CHAPTER 1

INTRODUCTION

The Real-Time Stock Price Alert System automatically monitors stock prices and notifies users when a specific stock reaches a predefined price threshold. This automation saves users the time of manually tracking prices. Upon reaching the target price, the system sends real-time alerts via SMS, email, or mobile notifications, enabling investors and traders to respond quickly to market fluctuations and seize opportunities.

CLOUD COMPUTING:
The system is hosted in a scalable, highly available cloud environment without relying on physical servers. Utilizing AWS's global infrastructure provides flexibility, scalability, and reliability. It automatically adjusts resources based on demand, managing usage fluctuations without downtime, simplifying deployment and maintenance, and reducing operational complexity.

SERVERLESS ARCHITECTURE:
Serverless architecture, primarily using AWS Lambda, is vital to the system. Functions execute only when triggered, minimizing the need for continuous server management and reducing costs by utilizing resources only as needed. AWS Lambda delivers high performance and low latency, providing immediate actions in response to real-time events.
The serverless model ensures that:

- The system can automatically scale with demand.
- There is no need to manage or update the underlying infrastructure.
- The system only incurs costs for actual usage, optimizing budget efficiency.
- Performance remains high, as resources are instantly allocated to handle alerts and notifications, ensuring users receive timely updates.

REAL-TIME DATA PROCESSING:
In the fast-moving stock market, real-time data processing is crucial. AWS serverless services manage stock data and trigger notifications based on preset conditions. Amazon EventBridge, integrated with CloudWatch, monitors data flow, ensuring instant reaction to stock price changes and timely alerts.

MESSAGING AND NOTIFICATION SYSTEMS:
Amazon Simple Notification Service (SNS) is used to send alerts through various channels, such as SMS and email. This flexibility enhances user experience by providing timely updates, while SNS guarantees high availability and reliable message delivery, keeping users informed and able to respond swiftly to market changes.

SCALABILITY AND FLEXIBILITY:
Scalability is a significant advantage of cloud infrastructure, allowing the system to adjust resources automatically as stock data volumes grow without manual intervention. AWS Cloud9 serves as the development platform, enabling collaborative real-time coding and improving the development process, allowing for quick adaptation to changing requirements.

CHAPTER 2
EXECUTIVE SUMMARY

The **Real-Time Stock Price Alert System** is an automated AWS-based solution that continuously monitors stock prices and sends alerts when predefined thresholds are reached. This eliminates manual tracking, helping investors make timely decisions in a fast-moving market. By using AWS Lambda, Amazon SNS, and CloudWatch, along with a third-party API like Alpha Vantage, the system provides real-time notifications via email or SMS. The cloud integration ensures high availability and reliability, simplifying stock monitoring for users.

Key Components:

- **Amazon SNS (Simple Notification Service):**
  Amazon SNS (Simple Notification Service) is a managed messaging service that decouples microservices and serverless applications. In your Real-Time Stock Price Alert System, SNS serves as the communication hub, sending alerts based on stock price thresholds through channels like SMS, email, and custom endpoints, ensuring real-time updates without delay.
- **AWS Lambda:**
  AWS Lambda is a serverless compute service that allows users to run code without managing servers. In your Real-Time Stock Price Alert System, Lambda analyzes stock prices and compares them against preset conditions, triggering notifications as needed. This ensures resources are used only when checking stock prices, leading to cost efficiency.
- **Amazon CloudWatch (with EventBridge):**
  Amazon CloudWatch is a monitoring service that provides real-time insights into system performance. In your Real-Time Stock Price Alert System, it continuously tracks stock prices, ensuring effective monitoring. Paired with EventBridge, it triggers Lambda functions when predefined stock conditions are met, enabling event-driven automation.
- **IAM (Identity and Access Management):**
  IAM (Identity and Access Management) is a service that controls access between different AWS services. In your Real-Time Stock Price Alert System, IAM ensures that only authorized services and users can access sensitive data or resources, such as those associated with Lambda, SNS, and CloudWatch.
- **AWS Cloud9:**
  AWS Cloud9 is an integrated development environment (IDE) that enables real-time coding and testing. In your Real-Time Stock Price Alert System, Cloud9 is used as the development environment, facilitating easy deployment of updates or patches to the system.

Use Cases:

- Investment Monitoring: Sends alerts when stock prices hit user-defined thresholds, aiding in timely decision-making.
- Risk Management: Helps users manage portfolio risks by notifying them of price drops.
- Day Trading: Provides real-time stock alerts for day traders to act on short-term fluctuations.
- Automated Trading: Can trigger buy or sell orders based on stock price changes.
- Watchlist Notifications: Keeps users informed about stocks on their watchlist.
- Educational Tool: Useful for teaching real-world stock tracking and alert systems.

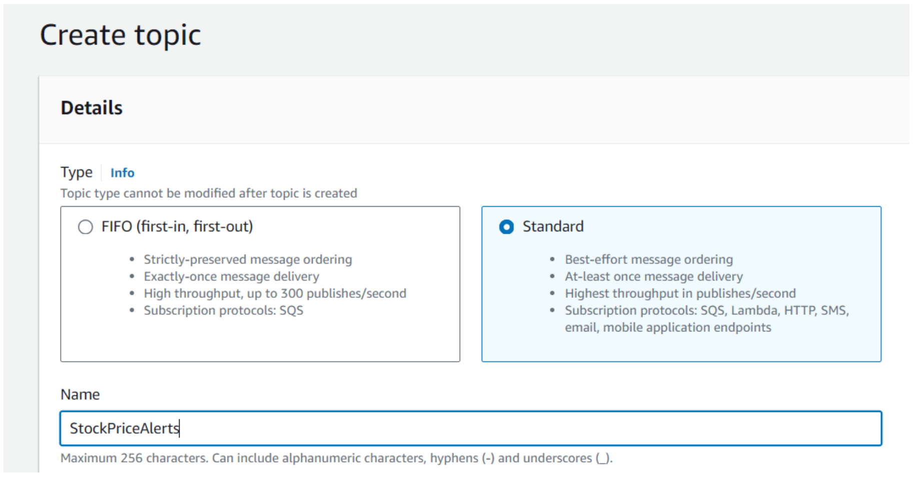
CHAPTER 3

STOCK PRICE MONITORING AND ALERT SYSTEM

Implementing a Stock Price Alert System that leverages the reliability and flexibility of AWS services, including Lambda for real-time price analysis, SNS for immediate notifications, and CloudWatch for system monitoring. By integrating these services, the system can effectively track stock prices, trigger alerts based on user-defined thresholds, and maintain optimal performance while ensuring robust security measures are in place. This comprehensive approach enables users to receive timely updates, empowering them to make informed investment decisions swiftly.

Steps Involved:

1. Access the Amazon SNS Console and Create a Topic
   - ➢ Log in to your AWS Management Console and search for SNS to select Simple Notification Service.
   - ➢ Click on Topics in the left navigation pane, then select Create Topic.
   - ➢ Choose Standard as the Type, and enter StockPriceAlerts as the Name.
   - ➢ Click “Create topic”.



Create topic

Details

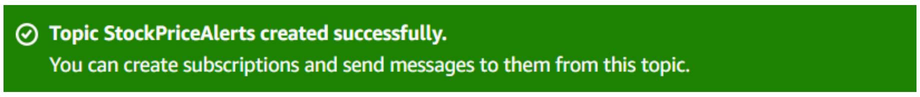
Type Info
Topic type cannot be modified after topic is created

○ FIFO (first-in, first-out)
- Strictly-preserved message ordering
- Exactly-once message delivery
- High throughput, up to 300 publishes/second
- Subscription protocols: SQS

● Standard
- Best-effort message ordering
- At-least once message delivery
- Highest throughput in publishes/second
- Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

Name
StockPriceAlerts
Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

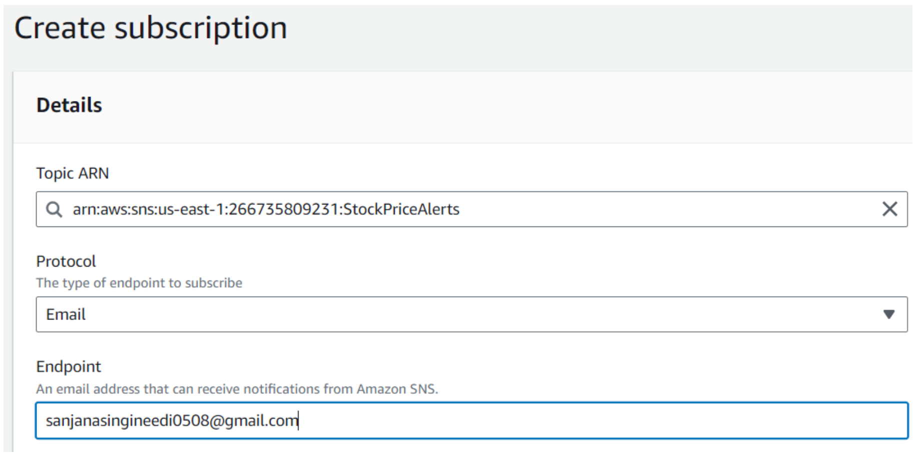
Figure 3.1 Create a Topic



Topic StockPriceAlerts created successfully.
You can create subscriptions and send messages to them from this topic.

Figure 3.2 Topic Successfully Created

2. Set Up a Subscription
   - ➢ Navigate to the Subscriptions section in the left navigation pane and click on Create Subscription.
   - ➢ Select the StockPriceAlerts topic from the Topic ARN dropdown.
   - ➢ Choose Email or SMS as the Protocol:
     - o For Email: Enter your email address (e.g., youremail@example.com).
     - o For SMS: Enter your phone number in the appropriate format (e.g., +11234567890).
   - ➢ Click Create subscription.



Create subscription

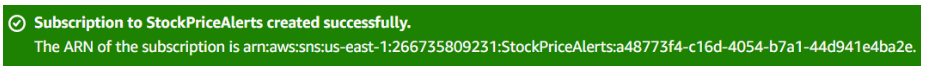
Details

Topic ARN: arn:aws:sns:us-east-1:266735809231:StockPriceAlerts

Protocol
The type of endpoint to subscribe
Email

Endpoint
An email address that can receive notifications from Amazon SNS.
sanjanasingineedi0508@gmail.com

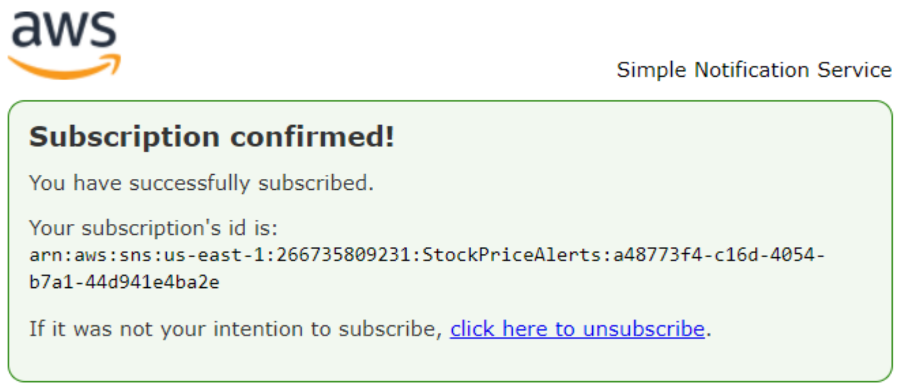
Figure 3.3 Create Subscription



Subscription to StockPriceAlerts created successfully.
The ARN of the subscription is arn:aws:sns:us-east-1:266735809231:StockPriceAlerts:a48773f4-c16d-4054-b7a1-44d941e4ba2e.

Figure 3.4 Subscription Successfully Created

3. Confirm the Subscription (for Email Alerts):
   - ➢ If you chose Email, check your inbox for a confirmation email from Amazon SNS.
   - ➢ Open the email and click on the confirmation link to activate your subscription.



Simple Notification Service

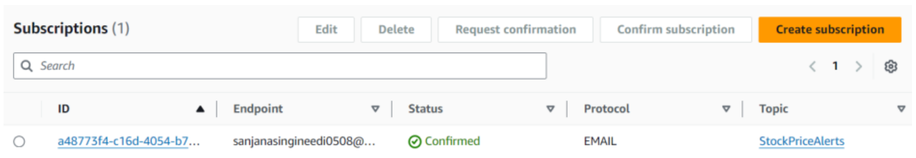
Subscription confirmed!

You have successfully subscribed.

Your subscription's id is:
arn:aws:sns:us-east-1:266735809231:StockPriceAlerts:a48773f4-c16d-4054-b7a1-44d941e4ba2e

If it was not your intention to subscribe, click here to unsubscribe.

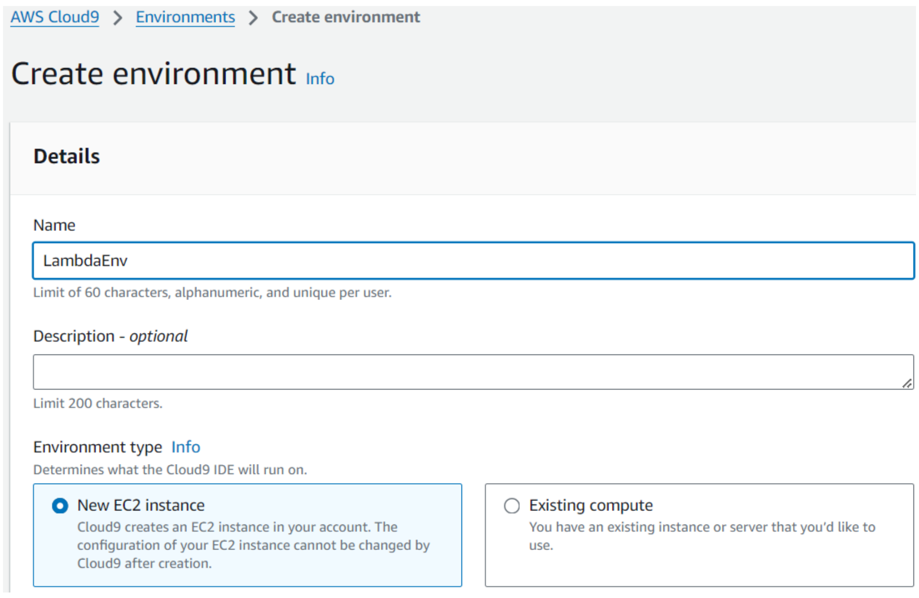
Figure 3.5 Subscribe through Endpoint



ID	Endpoint	Status	Protocol	Topic
a48773f4-c16d-4054-b7...	sanjanasingineedi0508@...	✓ Confirmed	EMAIL	StockPriceAlerts

Figure 3.6 Subscription Confirmed

4. Set Up AWS Cloud9 Environment
   - ➢ Search for Cloud9.
   - ➢ Click on Create environment.
   - ➢ Provide a name (e.g., LambdaEnv) and configure the instance as needed (e.g., select t2.micro as a default instance size).
   - ➢ Click Next, then click Create environment.


AWS Cloud9 > Environments > Create environment

Create environment Info

Details

Name

LambdaEnv

Limit of 60 characters, alphanumeric, and unique per user.

Description - *optional*

Limit 200 characters.

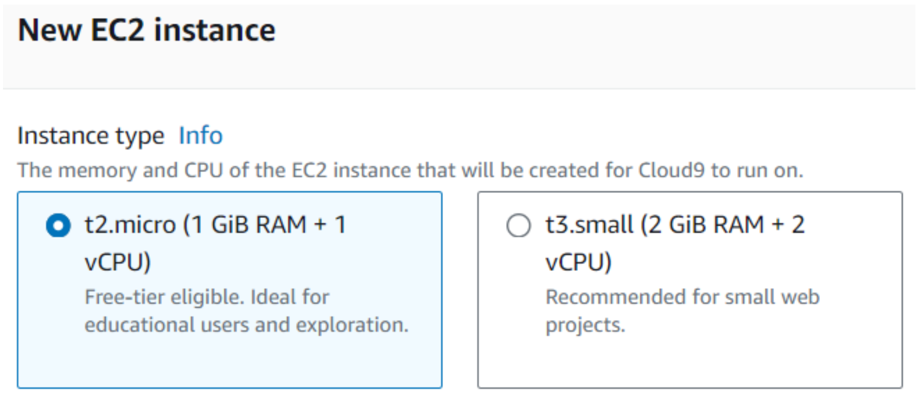
Environment type Info

Determines what the Cloud9 IDE will run on.

[✓ New EC2 instance] Cloud9 creates an EC2 instance in your account. The configuration of your EC2 instance cannot be changed by Cloud9 after creation.

Existing compute — You have an existing instance or server that you'd like to use.

Figure 3.7 Create Environment (Cloud9)


New EC2 instance

Instance type Info

The memory and CPU of the EC2 instance that will be created for Cloud9 to run on.

[✓ t2.micro (1 GiB RAM + 1 vCPU)] Free-tier eligible. Ideal for educational users and exploration.

t3.small (2 GiB RAM + 2 vCPU) — Recommended for small web projects.

Figure 3.8 Select the Instance Type (EC2 Instance)

5. Create a Directory for Your Lambda Function
   - Once your Cloud9 environment is set up, open a new terminal inside Cloud9.
   - Create a directory for your Lambda function by running
     Command:
     ```
     mkdir lambda_requests_function
     cd lambda_requests_function
     ```

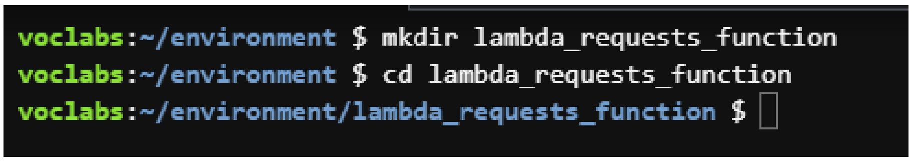
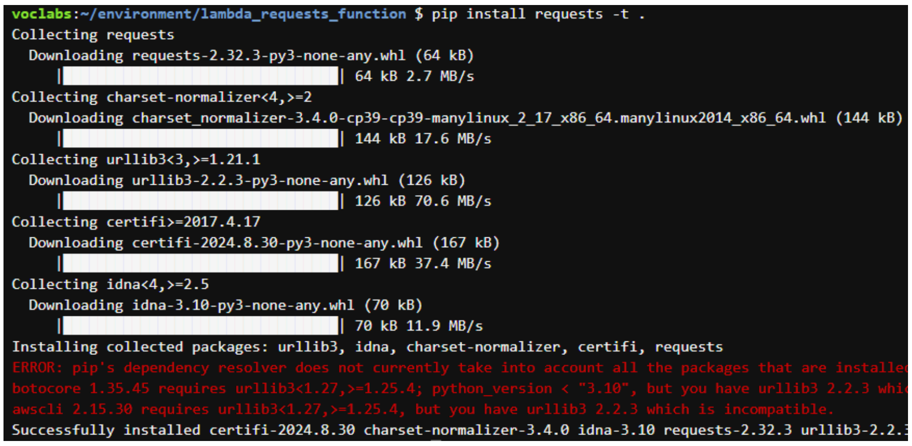

```
voclabs:~/environment $ mkdir lambda_requests_function
voclabs:~/environment $ cd lambda_requests_function
voclabs:~/environment/lambda_requests_function $ 
```

Figure 3.9 Create a Directory

6. Install the Requests Library
   - Install the requests library in the current directory by executing
     Command:
     ```
     pip install requests -t .
     ```

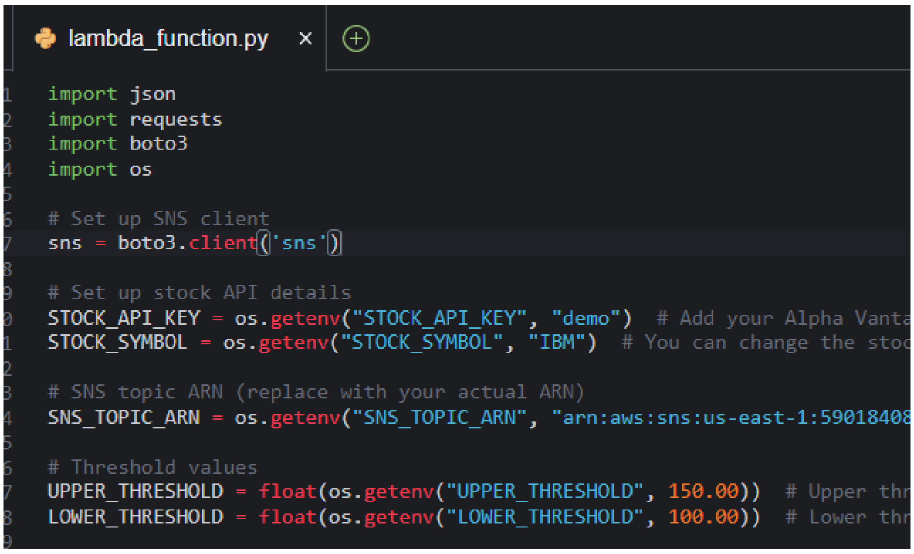


```
voclabs:~/environment/lambda_requests_function $ pip install requests -t .
Collecting requests
  Downloading requests-2.32.3-py3-none-any.whl (64 kB)
     |████████████████████████████████| 64 kB 2.7 MB/s
Collecting charset-normalizer<4,>=2
  Downloading charset_normalizer-3.4.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (144 kB)
     |████████████████████████████████| 144 kB 17.6 MB/s
Collecting urllib3<3,>=1.21.1
  Downloading urllib3-2.2.3-py3-none-any.whl (126 kB)
     |████████████████████████████████| 126 kB 70.6 MB/s
Collecting certifi>=2017.4.17
  Downloading certifi-2024.8.30-py3-none-any.whl (167 kB)
     |████████████████████████████████| 167 kB 37.4 MB/s
Collecting idna<4,>=2.5
  Downloading idna-3.10-py3-none-any.whl (70 kB)
     |████████████████████████████████| 70 kB 11.9 MB/s
Installing collected packages: urllib3, idna, charset-normalizer, certifi, requests
ERROR: pip's dependency resolver does not currently take into account all the packages that are installe
botocore 1.35.45 requires urllib3<1.27,>=1.25.4; python_version < "3.10", but you have urllib3 2.2.3 whi
awscli 2.15.30 requires urllib3<1.27,>=1.25.4, but you have urllib3 2.2.3 which is incompatible.
Successfully installed certifi-2024.8.30 charset-normalizer-3.4.0 idna-3.10 requests-2.32.3 urllib3-2.2.
```

Figure 3.10 Install requests Package

- ➢ This command installs the requests module and its dependencies into the lambda_requests_function directory.

7. Add Your Lambda Function Code
   - ➢ In the Cloud9 file explorer, create a new Python file named lambda_function.py in the lambda_requests_function directory.
   - ➢ Add your Lambda function code to this file.



```
lambda_function.py  ×

import json
import requests
import boto3
import os

# Set up SNS client
sns = boto3.client('sns')

# Set up stock API details
STOCK_API_KEY = os.getenv("STOCK_API_KEY", "demo")  # Add your Alpha Vanta
STOCK_SYMBOL = os.getenv("STOCK_SYMBOL", "IBM")  # You can change the stoc

# SNS topic ARN (replace with your actual ARN)
SNS_TOPIC_ARN = os.getenv("SNS_TOPIC_ARN", "arn:aws:sns:us-east-1:59018408

# Threshold values
UPPER_THRESHOLD = float(os.getenv("UPPER_THRESHOLD", 150.00))  # Upper thr
LOWER_THRESHOLD = float(os.getenv("LOWER_THRESHOLD", 100.00))  # Lower thr
```

Figure 3.11 Create lambda_function.py file

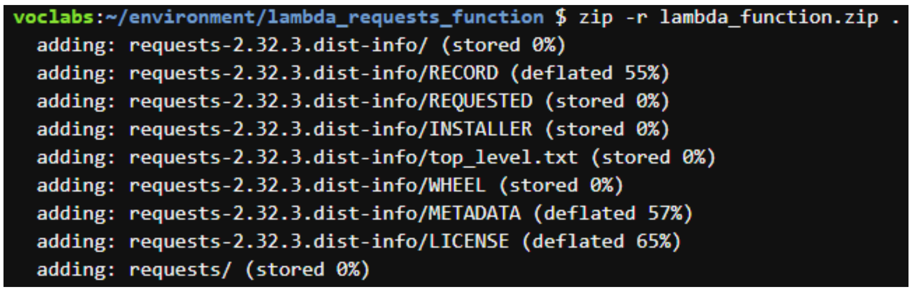
Program:

```
import json
import requests
import boto3
import os

# Set up SNS client
sns = boto3.client('sns')

# Set up stock API details
STOCK_API_KEY = os.getenv("STOCK_API_KEY", "demo")
```

```
STOCK_SYMBOL = os.getenv("STOCK_SYMBOL", "IBM")

# SNS topic ARN (replace with your actual ARN)
SNS_TOPIC_ARN = os.getenv("SNS_TOPIC_ARN", "arn:aws:sns:us-
                      east-1:590184086939:StockPriceAlerts")

# Threshold values
UPPER_THRESHOLD = float(os.getenv("UPPER_THRESHOLD", 150.00))
LOWER_THRESHOLD = float(os.getenv("LOWER_THRESHOLD", 100.00))

def get_stock_price(symbol):
# Alpha Vantage API URL
url =
      f"https://www.alphavantage.co/query?function=TIME_SERIES_
      INTRADAY&symbol={STOCK_API_KEY}&interval=1min&apikey={STO
      CK_TOPIC_ARN}"
response = requests.get(url)
data = response.json()

# Extract the latest stock price
time_series = data.get("Time Series (1min)", {})
if not time_series:
return None

latest_time = sorted(time_series.keys())[0]
latest_price = float(time_series[latest_time]["1. open"])

return latest_price

def lambda_handler(event, context):
# Get the current stock price
stock_price = get_stock_price(STOCK_SYMBOL)

if stock_price is None:
print("Could not fetch stock price")
return {"statusCode": 500, "body": json.dumps("Stock
                                      price fetch failed")}

print(f"Current {STOCK_SYMBOL} price: {stock_price}")

# Check if the stock price exceeds the upper threshold
if stock_price > UPPER_THRESHOLD:
message = f"The stock price of {STOCK_SYMBOL} is
                ${stock_price}, which exceeds your upper
                threshold of ${UPPER_THRESHOLD}!"

# Send an SNS notification for upper threshold
sns.publish(
TopicArn=SNS_TOPIC_ARN,
Message=message,
Subject=f"{STOCK_SYMBOL} Stock Price Alert: Above
                                                 Threshold"
)
print("Upper threshold alert sent!")
```

```
# Check if the stock price falls below the lower threshold
elif stock_price < LOWER_THRESHOLD:
message = f"The stock price of {STOCK_SYMBOL} is
                 ${stock_price}, which is below your lower
                 threshold of ${LOWER_THRESHOLD}!"

# Send an SNS notification for lower threshold
sns.publish(
TopicArn=SNS_TOPIC_ARN,
Message=message,
Subject=f"{STOCK_SYMBOL} Stock Price Alert: Below
                                                 Threshold"
)
print("Lower threshold alert sent!")

else:
print(f"The stock price of {STOCK_SYMBOL} is within the
                                            threshold range.")

return {"statusCode": 200, "body": json.dumps(f"Checked
           stock price: ${stock_price}")}
```

8. Zip the Directory
   - After adding your code, zip the contents of the directory by running
     Command:
     ```
     zip -r lambda_function.zip .
     ```

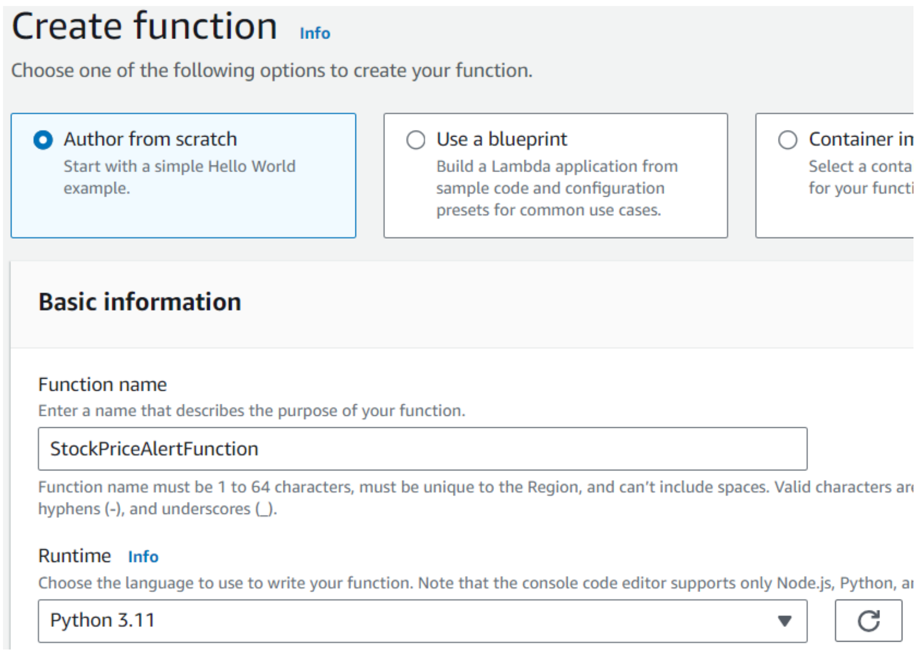


```
voclabs:~/environment/lambda_requests_function $ zip -r lambda_function.zip .
  adding: requests-2.32.3.dist-info/ (stored 0%)
  adding: requests-2.32.3.dist-info/RECORD (deflated 55%)
  adding: requests-2.32.3.dist-info/REQUESTED (stored 0%)
  adding: requests-2.32.3.dist-info/INSTALLER (stored 0%)
  adding: requests-2.32.3.dist-info/top_level.txt (stored 0%)
  adding: requests-2.32.3.dist-info/WHEEL (stored 0%)
  adding: requests-2.32.3.dist-info/METADATA (deflated 57%)
  adding: requests-2.32.3.dist-info/LICENSE (deflated 65%)
  adding: requests/ (stored 0%)
```

Figure 3.12 Zip the directory files

   - This creates a file named lambda_function.zip, containing both the requests library and your lambda_function.py code.
   - download the project file into the local system, extract the downloaded file since it is in the format of zip.

9. Access the AWS Lambda Console and Create a Function
   - In to your AWS Management Console and search for Lambda to select the AWS Lambda Console.
   - Click on Create Function, then choose Author from scratch.
   - Fill in the following details:
     - Function name: StockPriceAlertFunction
     - Runtime: Python 3.x
     - Execution role: Select Create a new role with basic Lambda permissions.
   - Click "Create function".



Create function Info

Choose one of the following options to create your function.

[✓ Author from scratch] Start with a simple Hello World example.

Use a blueprint — Build a Lambda application from sample code and configuration presets for common use cases.

Container in — Select a conta for your functi

Basic information

Function name: StockPriceAlertFunction

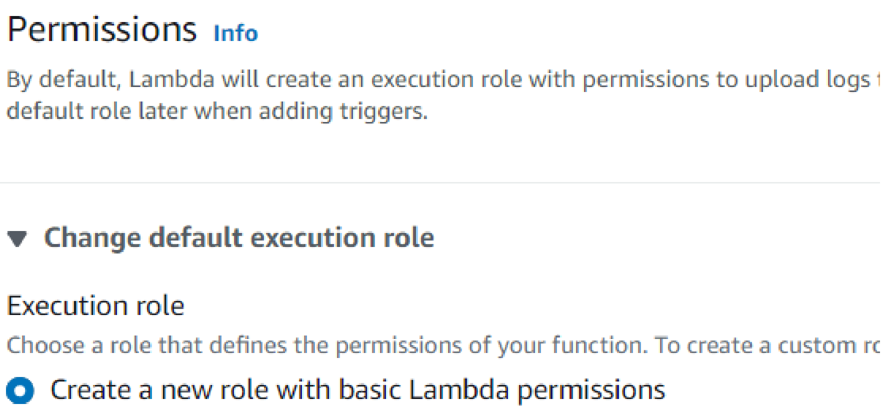
Enter a name that describes the purpose of your function.

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters ar hyphens (-), and underscores (_).

Runtime Info: Python 3.11

Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, a

Figure 3.13 Create function



Permissions Info

By default, Lambda will create an execution role with permissions to upload logs t default role later when adding triggers.

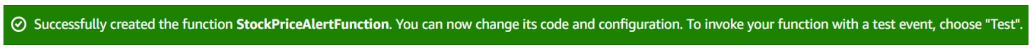
▼ Change default execution role

Execution role

Choose a role that defines the permissions of your function. To create a custom ro

[✓ Create a new role with basic Lambda permissions]

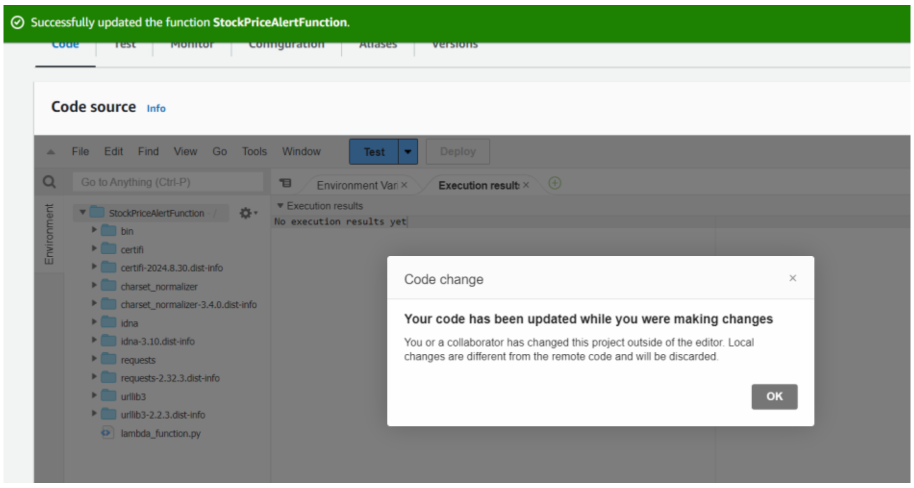
Figure 3.14 Select Execution role



Successfully created the function **StockPriceAlertFunction**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Figure 3.15 Successfully Created Function

10. Upload the ZIP File to AWS Lambda
    - ➢ Open the AWS Lambda Console in your browser.
    - ➢ Go to your existing Lambda function.
    - ➢ In the Code section, click Upload and choose the lambda_function.zip file you created.
    - ➢ Click Save to finalize the upload.


Successfully updated the function **StockPriceAlertFunction.**

Code source Info

Execution results

```
No execution results yet
```

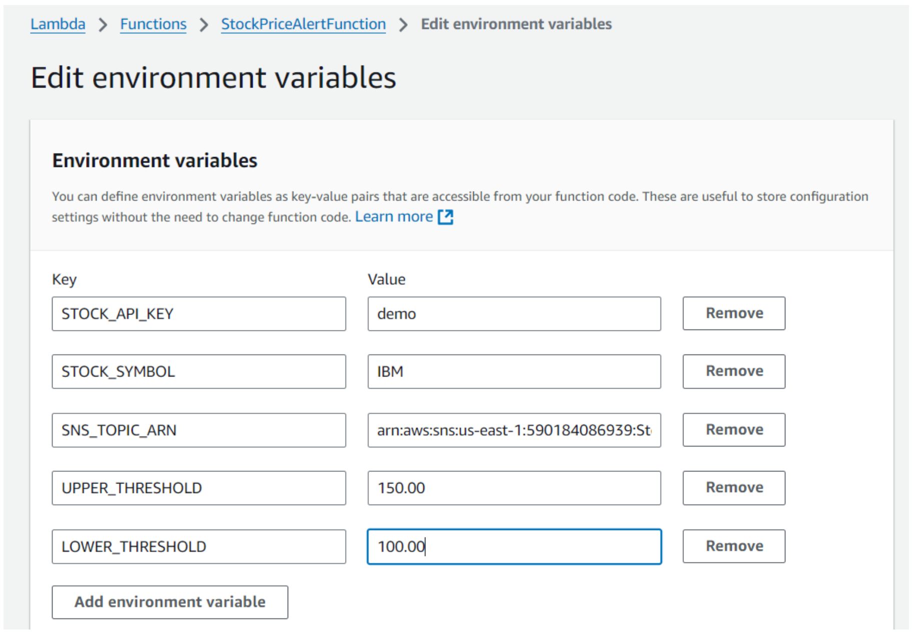
Code change

Your code has been updated while you were making changes

You or a collaborator has changed this project outside of the editor. Local changes are different from the remote code and will be discarded.

Figure 3.16 Upload the zip file to Lambda Console

11. Set Environment Variables in Lambda
   - ➢ Scroll down to the **Environment Variables** section.
   - ➢ Click **Edit** and add the following key-value pairs:
     - o **STOCK_API_KEY:** Your Alpha Vantage API key.
     - o **STOCK_SYMBOL:** The stock symbol you want to monitor (e.g., AAPL).
     - o **SNS_TOPIC_ARN:** The ARN of the SNS topic created earlier for notifications.
     - o **PRICE_THRESHOLD:** The stock price threshold that triggers alerts (both UPPER_THRESHOLD, LOWER_THRESHOLD)
   - ➢ Click **Save** to apply the changes.


Lambda > Functions > StockPriceAlertFunction > Edit environment variables

Edit environment variables

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. Learn more

Key	Value	
STOCK_API_KEY	demo	Remove
STOCK_SYMBOL	IBM	Remove
SNS_TOPIC_ARN	arn:aws:sns:us-east-1:590184086939:St	Remove
UPPER_THRESHOLD	150.00	Remove
LOWER_THRESHOLD	100.00	Remove

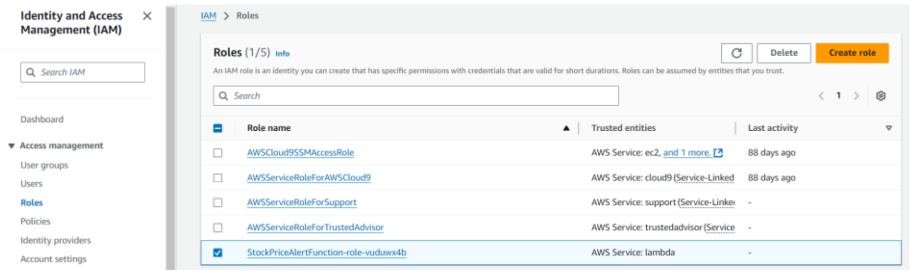
Figure 3.17 Edit Environment Variables

Successfully updated the function **StockPriceAlertFunction.**

Figure 3.18 Successfully updated the function

12. Set Lambda Permissions for SNS

- ➢ In the AWS IAM Console, go to Roles.

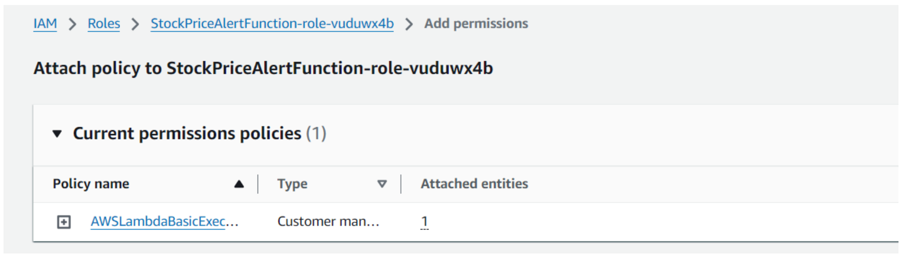


An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

	Role name	Trusted entities	Last activity
☐	AWSCloud9SSMAccessRole	AWS Service: ec2, and 1 more.	88 days ago
☐	AWSServiceRoleForAWSCloud9	AWS Service: cloud9 (Service-Linked	88 days ago
☐	AWSServiceRoleForSupport	AWS Service: support (Service-Linke	-
☐	AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service	-
☑	StockPriceAlertFunction-role-vuduwx4b	AWS Service: lambda	-

Figure 3.19 IAM roles

- ➢ Locate and select the execution role associated with your Lambda function.



IAM > Roles > StockPriceAlertFunction-role-vuduwx4b > Add permissions

Attach policy to StockPriceAlertFunction-role-vuduwx4b

Current permissions policies (1)

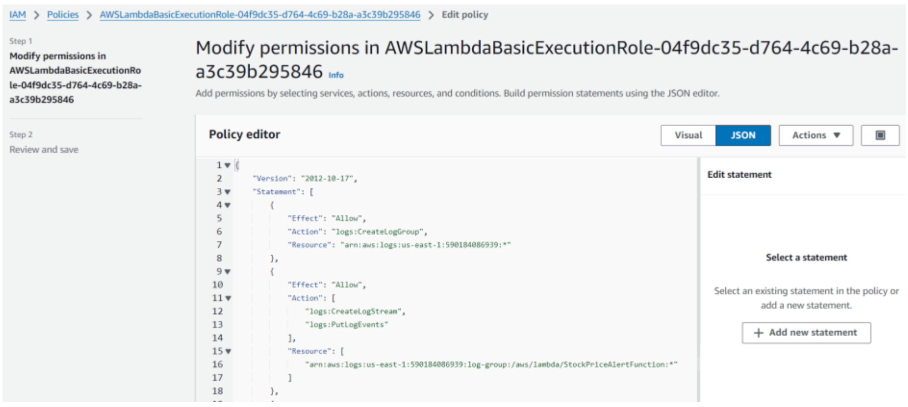
Policy name	Type	Attached entities
AWSLambdaBasicExec...	Customer man...	1

Figure 3.20 Lambda function Add Permissions

- ➢ Click Attach Policies and create a custom inline policy with the following permissions to allow Lambda to publish to SNS

  Code:

```
{
        "Version": "2012-10-17",
        "Statement": [
                {
                        "Effect": "Allow",
                        "Action": "logs:CreateLogGroup",
                        "Resource":"arn:aws:logs:us-east-
                                                    1:590184086939:*"
                },
                {
                        "Effect": "Allow",
                        "Action": [
                                "logs:CreateLogStream",
                                "logs:PutLogEvents"
                        ],
                        "Resource": [
                                "arn:aws:logs:us-east-1:590184086939:log-
                        group:/aws/lambda/StockPriceAlertFunction:*"
                        ]
                },
                {
                        "Effect": "Allow",
                        "Action": "sns:Publish",
                        "Resource": "arn:aws:sns:us-east-
                                1:590184086939:StockPriceAlerts"
                }
        ]
}
```



IAM > Policies > AWSLambdaBasicExecutionRole-04f9dc35-d764-4c69-b28a-a3c39b295846 > Edit policy

Step 1
Modify permissions in AWSLambdaBasicExecutionRole-04f9dc35-d764-4c69-b28a-a3c39b295846

Step 2
Review and save

Modify permissions in AWSLambdaBasicExecutionRole-04f9dc35-d764-4c69-b28a-a3c39b295846

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

Policy editor

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "arn:aws:logs:us-east-1:590184086939:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:590184086939:log-group:/aws/lambda/StockPriceAlertFunction:*"
            ]
        },
```

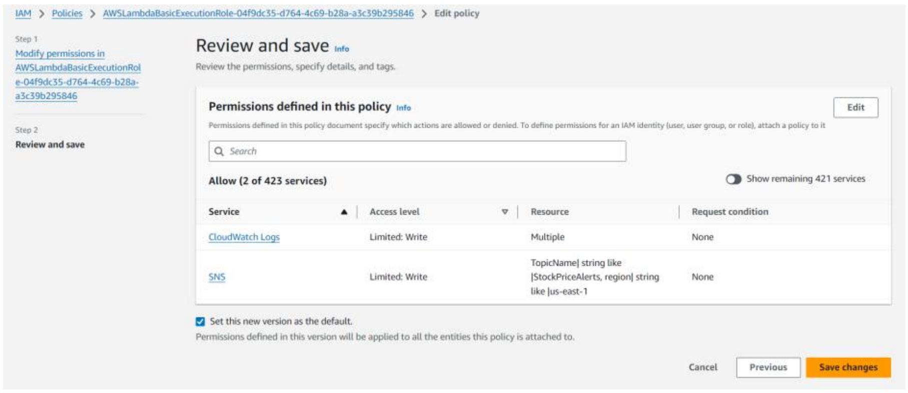
Edit statement

Select a statement

Select an existing statement in the policy or add a new statement.

Figure 3.21 Modify Permissions of Lambda Function

- ➢ Review and save the IAM role Policy



IAM > Policies > AWSLambdaBasicExecutionRole-04f9dc35-d764-4c69-b28a-a3c39b295846 > Edit policy

Step 1
Modify permissions in AWSLambdaBasicExecutionRole-04f9dc35-d764-4c69-b28a-a3c39b295846

Step 2
Review and save

Review and save

Review the permissions, specify details, and tags.

Permissions defined in this policy

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

Allow (2 of 423 services)

Show remaining 421 services

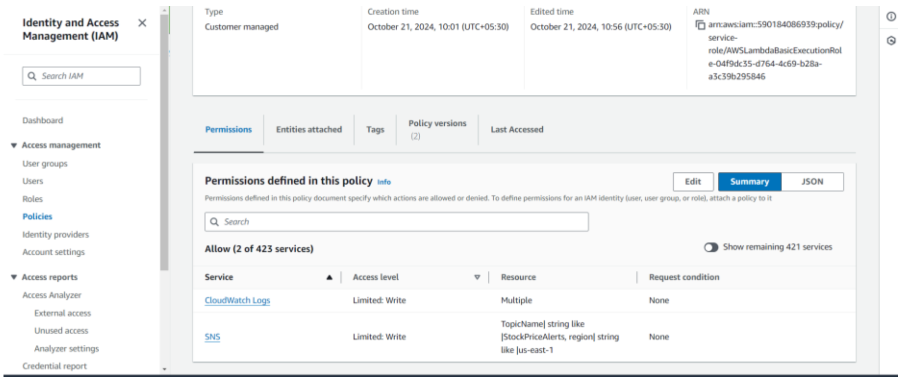
Service	Access level	Resource	Request condition
CloudWatch Logs	Limited: Write	Multiple	None
SNS	Limited: Write	TopicName| string like |StockPriceAlerts, region| string like |us-east-1	None

☑ Set this new version as the default.
Permissions defined in this version will be applied to all the entities this policy is attached to.

Figure 3.22 Review and Save

Policy AWSLambdaBasicExecutionRole-04f9dc35-d764-4c69-b28a-a3c39b295846 updated.

Figure 3.23 Policy Updated



Identity and Access Management (IAM)

Type	Creation time	Edited time	ARN
Customer managed	October 21, 2024, 10:01 (UTC+05:30)	October 21, 2024, 10:56 (UTC+05:30)	arn:aws:iam::590184086939:policy/service-role/AWSLambdaBasicExecutionRole-04f9dc35-d764-4c69-b28a-a3c39b295846

Permissions defined in this policy

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

Allow (2 of 423 services)

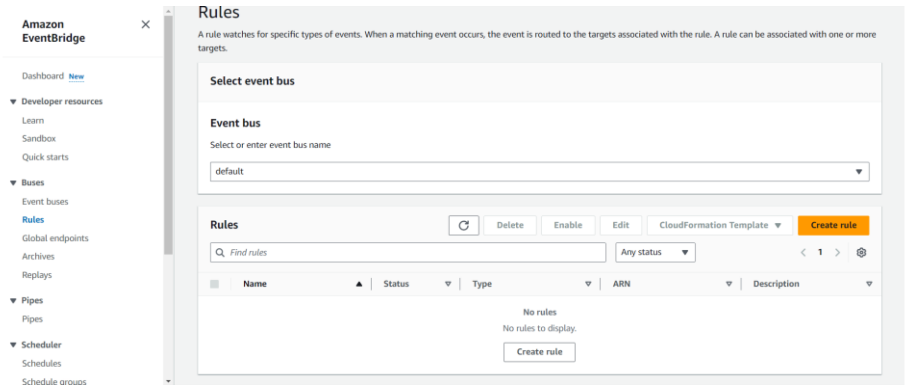
Show remaining 421 services

Service	Access level	Resource	Request condition
CloudWatch Logs	Limited: Write	Multiple	None
SNS	Limited: Write	TopicName| string like |StockPriceAlerts, region| string like |us-east-1	None

Figure 3.24 Permissions defined in the Policy

13. Schedule the Lambda Function with CloudWatch
    - ➢ Go to the Amazon EventBridge console and select "Create Rule."
    - ➢ Name the rule, optionally add a description, and choose the default event bus.
    - ➢ Select "Schedule" as the rule type for setting a recurring or one-time event.
    - ➢ Set the schedule and choose the Lambda function as the target.

➢ Review the rule details and create it to activate the schedule.


Amazon EventBridge
Dashboard New
Developer resources
Learn
Sandbox
Quick starts
Buses
Event buses
Rules
Global endpoints
Archives
Replays
Pipes
Pipes
Scheduler
Schedules
Schedule groups

Rules
A rule watches for specific types of events. When a matching event occurs, the event is routed to the targets associated with the rule. A rule can be associated with one or more targets.

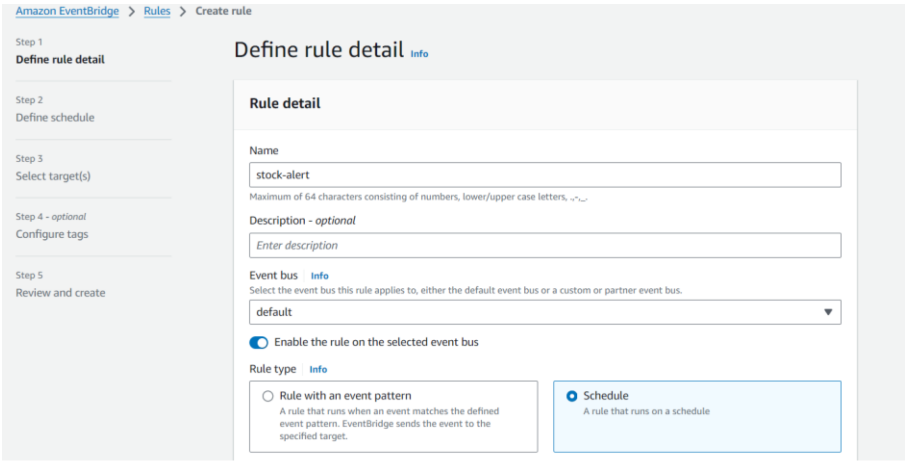
Select event bus
Event bus
Select or enter event bus name
default

Rules
Delete | Enable | Edit | CloudFormation Template | Create rule
Find rules | Any status

Name	Status	Type	ARN	Description

No rules
No rules to display.
Create rule

Figure 3.25 Create Rule

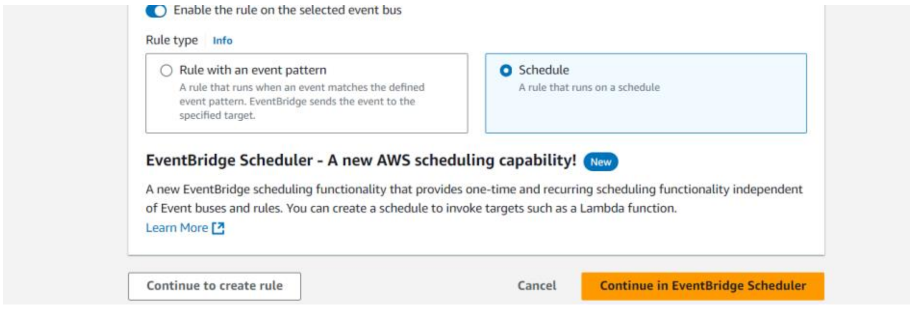

Amazon EventBridge > Rules > Create rule

Step 1
Define rule detail
Step 2
Define schedule
Step 3
Select target(s)
Step 4 - *optional*
Configure tags
Step 5
Review and create

Define rule detail Info

Rule detail
Name
stock-alert
Maximum of 64 characters consisting of numbers, lower/upper case letters, .,-,_.
Description - *optional*
Enter description
Event bus Info
Select the event bus this rule applies to, either the default event bus or a custom or partner event bus.
default
Enable the rule on the selected event bus
Rule type Info
Rule with an event pattern
A rule that runs when an event matches the defined event pattern. EventBridge sends the event to the specified target.
Schedule
A rule that runs on a schedule

Figure 3.26 Define rule Detail

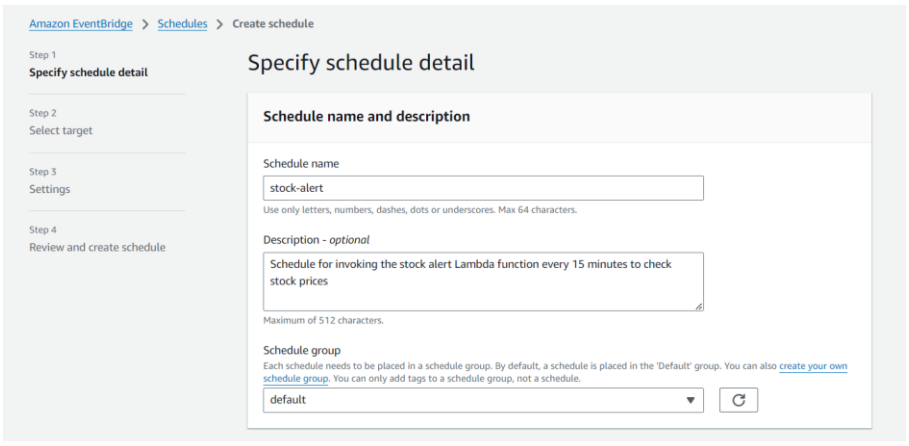

Enable the rule on the selected event bus
Rule type Info
Rule with an event pattern
A rule that runs when an event matches the defined event pattern. EventBridge sends the event to the specified target.
Schedule
A rule that runs on a schedule

EventBridge Scheduler - A new AWS scheduling capability! New
A new EventBridge scheduling functionality that provides one-time and recurring scheduling functionality independent of Event buses and rules. You can create a schedule to invoke targets such as a Lambda function.
Learn More

Continue to create rule | Cancel | Continue in EventBridge Scheduler

Figure 3.27 Create Schedule

➢ Set the event source:
  - Event Source: Choose EventBridge (CloudWatch Events).
  - Rule Type: Select Schedule expression.
  - Schedule Expression: Enter rate(15 minutes) to trigger the function every 15 minutes.


Amazon EventBridge > Schedules > Create schedule

Step 1
Specify schedule detail

Step 2
Select target

Step 3
Settings

Step 4
Review and create schedule

Specify schedule detail

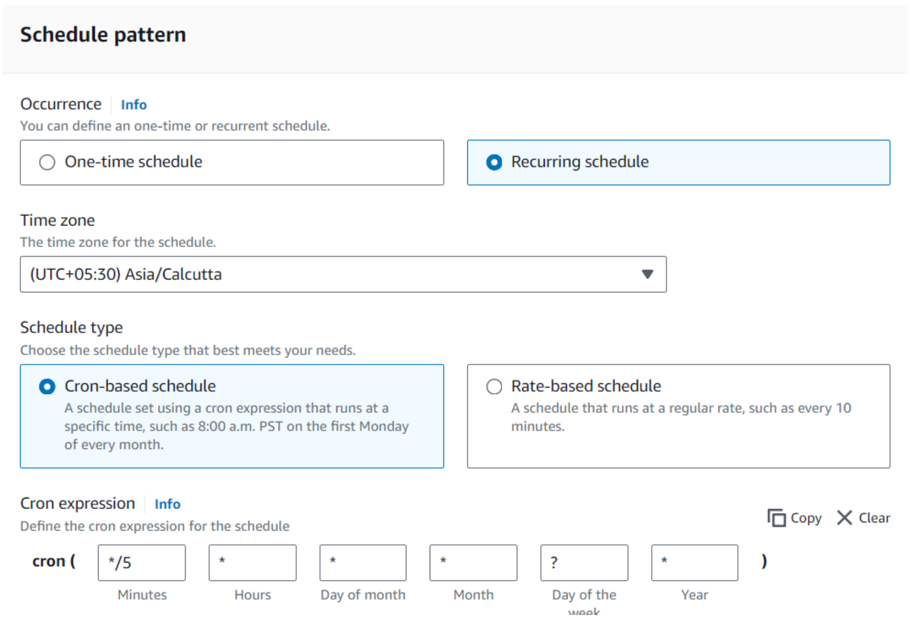
Schedule name and description

Schedule name
stock-alert
Use only letters, numbers, dashes, dots or underscores. Max 64 characters.

Description - *optional*
Schedule for invoking the stock alert Lambda function every 15 minutes to check stock prices
Maximum of 512 characters.

Schedule group
Each schedule needs to be placed in a schedule group. By default, a schedule is placed in the 'Default' group. You can also create your own schedule group. You can only add tags to a schedule group, not a schedule.
default

Figure 3.28 Specify Schedule detail


Schedule pattern

Occurrence Info
You can define an one-time or recurrent schedule.
One-time schedule [✓ Recurring schedule]

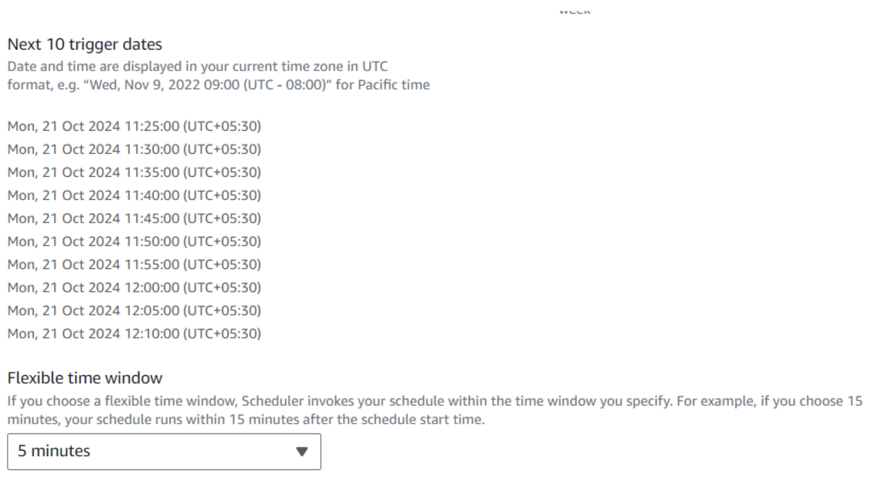
Time zone
The time zone for the schedule.
(UTC+05:30) Asia/Calcutta

Schedule type
Choose the schedule type that best meets your needs.
[✓ Cron-based schedule]
A schedule set using a cron expression that runs at a specific time, such as 8:00 a.m. PST on the first Monday of every month.
Rate-based schedule
A schedule that runs at a regular rate, such as every 10 minutes.

Cron expression Info
Define the cron expression for the schedule
Copy Clear

cron (	*/5	*	*	*	?	*	)
	Minutes	Hours	Day of month	Month	Day of the week	Year	

Figure 3.29 Define Schedule Pattern

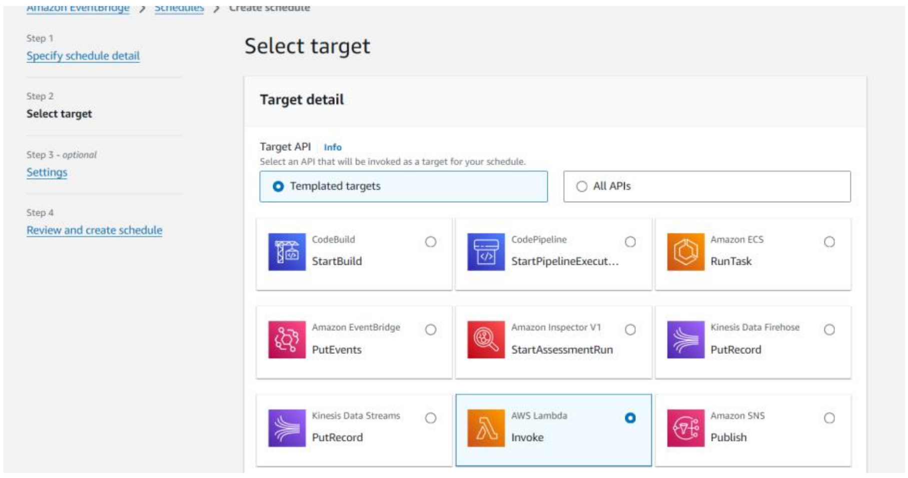

Next 10 trigger dates
Date and time are displayed in your current time zone in UTC format, e.g. "Wed, Nov 9, 2022 09:00 (UTC - 08:00)" for Pacific time

Mon, 21 Oct 2024 11:25:00 (UTC+05:30)
Mon, 21 Oct 2024 11:30:00 (UTC+05:30)
Mon, 21 Oct 2024 11:35:00 (UTC+05:30)
Mon, 21 Oct 2024 11:40:00 (UTC+05:30)
Mon, 21 Oct 2024 11:45:00 (UTC+05:30)
Mon, 21 Oct 2024 11:50:00 (UTC+05:30)
Mon, 21 Oct 2024 11:55:00 (UTC+05:30)
Mon, 21 Oct 2024 12:00:00 (UTC+05:30)
Mon, 21 Oct 2024 12:05:00 (UTC+05:30)
Mon, 21 Oct 2024 12:10:00 (UTC+05:30)

Flexible time window
If you choose a flexible time window, Scheduler invokes your schedule within the time window you specify. For example, if you choose 15 minutes, your schedule runs within 15 minutes after the schedule start time.
5 minutes

Figure 3.30 Trigger Dates & Time window

- Add the Lambda function as the target:
  - Under Target, choose Lambda Function.
  - Select your StockPriceAlertFunction from the dropdown.


Step 1
Specify schedule detail

Step 2
Select target

Step 3 - *optional*
Settings

Step 4
Review and create schedule

Select target

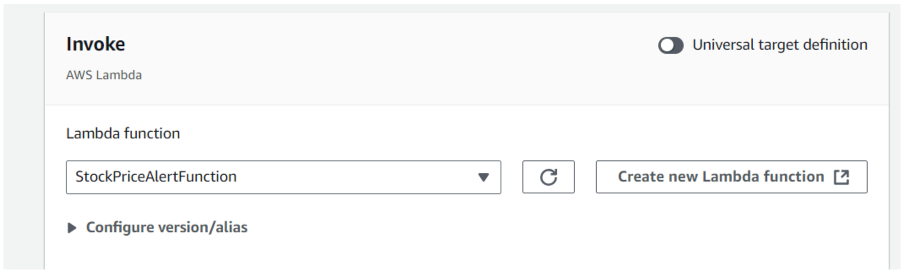
Target detail

Target API Info
Select an API that will be invoked as a target for your schedule.

[✓ Templated targets] All APIs

CodeBuild StartBuild	CodePipeline StartPipelineExecut...	Amazon ECS RunTask
Amazon EventBridge PutEvents	Amazon Inspector V1 StartAssessmentRun	Kinesis Data Firehose PutRecord
Kinesis Data Streams PutRecord	[✓ AWS Lambda Invoke]	Amazon SNS Publish

Figure 3.31 Select Target


Invoke
AWS Lambda

Universal target definition

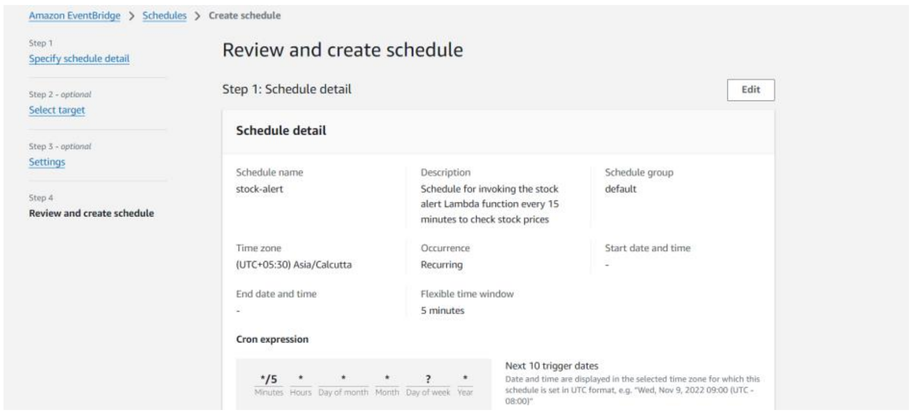
Lambda function
StockPriceAlertFunction

Create new Lambda function

Configure version/alias

Figure 3.32 Invoke (Lambda function)

- ➢ Review and create the rule.



Step 1
Specify schedule detail

Step 2 - *optional*
Select target

Step 3 - *optional*
Settings

Step 4
Review and create schedule

Review and create schedule

Step 1: Schedule detail

Schedule detail

Schedule name	Description	Schedule group
stock-alert	Schedule for invoking the stock alert Lambda function every 15 minutes to check stock prices	default

Time zone	Occurrence	Start date and time
(UTC+05:30) Asia/Calcutta	Recurring	-

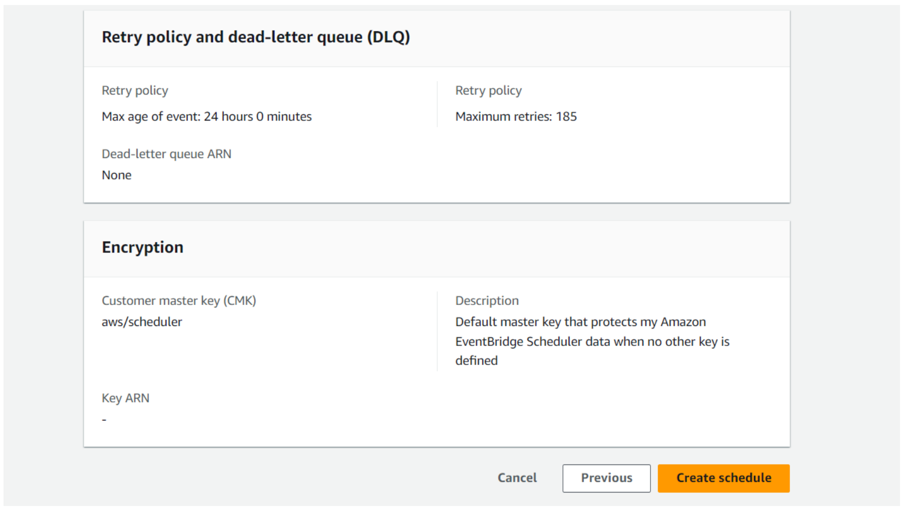
End date and time	Flexible time window
-	5 minutes

Cron expression

Minutes	Hours	Day of month	Month	Day of week	Year
*/5	*	*	*	?	*

Next 10 trigger dates
Date and time are displayed in the selected time zone for which this schedule is set in UTC format, e.g. "Wed, Nov 9, 2022 09:00 (UTC - 08:00)"

Figure 3.33 Review Schedule


Retry policy and dead-letter queue (DLQ)

Retry policy
Max age of event: 24 hours 0 minutes

Retry policy
Maximum retries: 185

Dead-letter queue ARN
None

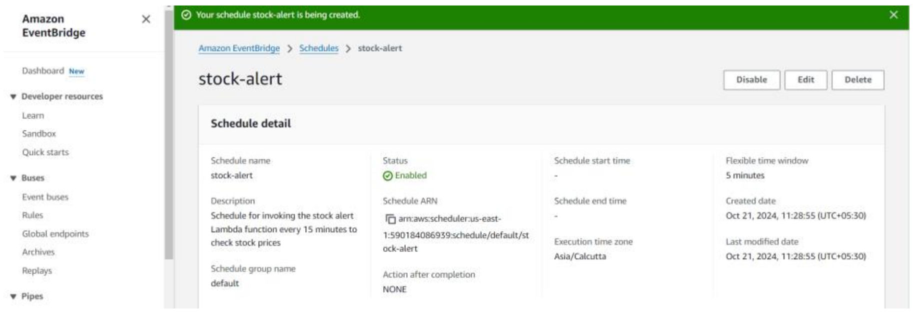
Encryption

Customer master key (CMK)
aws/scheduler

Description
Default master key that protects my Amazon EventBridge Scheduler data when no other key is defined

Key ARN
-

Figure 3.34 Create Schedule



Your schedule stock-alert is being created.

stock-alert

Schedule detail

Schedule name
stock-alert

Description
Schedule for invoking the stock alert Lambda function every 15 minutes to check stock prices

Schedule group name
default

Status
Enabled

Schedule ARN
arn:aws:scheduler:us-east-1:590184086939:schedule/default/stock-alert

Action after completion
NONE

Schedule start time
-

Schedule end time
-

Execution time zone
Asia/Calcutta

Flexible time window
5 minutes

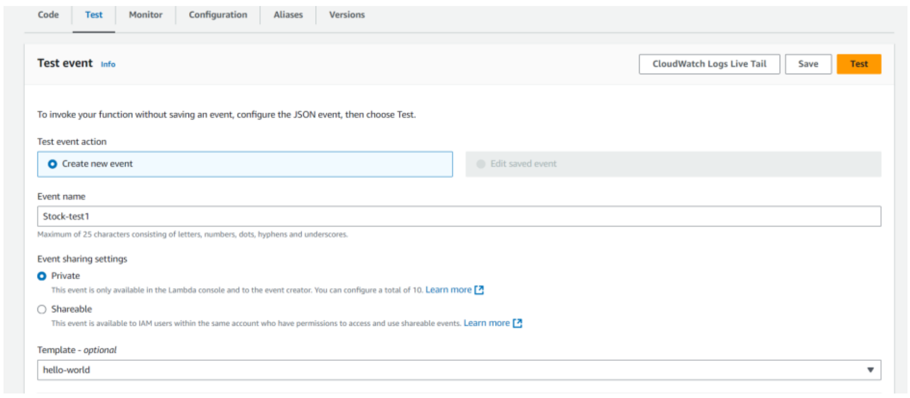
Created date
Oct 21, 2024, 11:28:55 (UTC+05:30)

Last modified date
Oct 21, 2024, 11:28:55 (UTC+05:30)

Figure 3.35 Schedule Successfully Created

14. Create a Test Event for Lambda Function
    - ➢ In the AWS Lambda Console, navigate to your Lambda function and click on the Test button at the top right of the function's detail page.
    - ➢ If this is your first test, you'll be prompted to Configure test events.
        - o Select Create new test event.
        - o Name your event, for example, TestEvent.
        - o Use the following simple JSON structure for the test event

    Code:
```
{
        "STOCK_API_KEY": "demo",
        "STOCK_SYMBOLS": "IBM",
        "SNS_TOPIC_ARN": "arn:aws:sns:us-east-
                             1:590184086939:StockPriceAlerts",
        "UPPER_THRESHOLD": 150.00,
        "LOWER_THRESHOLD": 100.00,
        "message": "Check stock price for IBM Inc."
}
```



Test event Info

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action: [✓ Create new event] Edit saved event

Event name: Stock-test1

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

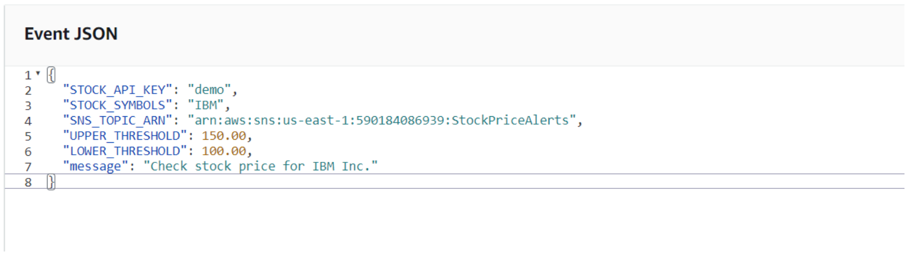
Event sharing settings

[✓ Private] This event is only available in the Lambda console and to the event creator. You can configure a total of 10. Learn more

Shareable This event is available to IAM users within the same account who have permissions to access and use shareable events. Learn more

Template - *optional*: hello-world

Figure 3.36 Create Test Event



Event JSON

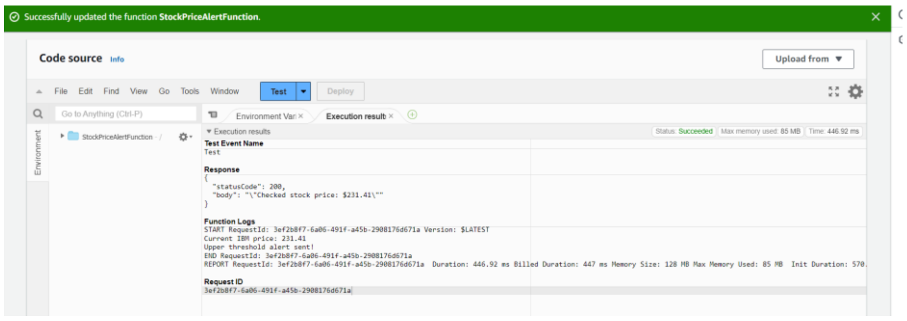
```
{
  "STOCK_API_KEY": "demo",
  "STOCK_SYMBOLS": "IBM",
  "SNS_TOPIC_ARN": "arn:aws:sns:us-east-1:590184086939:StockPriceAlerts",
  "UPPER_THRESHOLD": 150.00,
  "LOWER_THRESHOLD": 100.00,
  "message": "Check stock price for IBM Inc."
}
```

Figure 3.37 Event JSON

➢ Click Create and then Test to trigger the Lambda function using this event.

15. Test the System

➢ **Manual Test:** In the Lambda console, you can manually test your function by clicking the Test button after configuring a test event, as explained earlier.

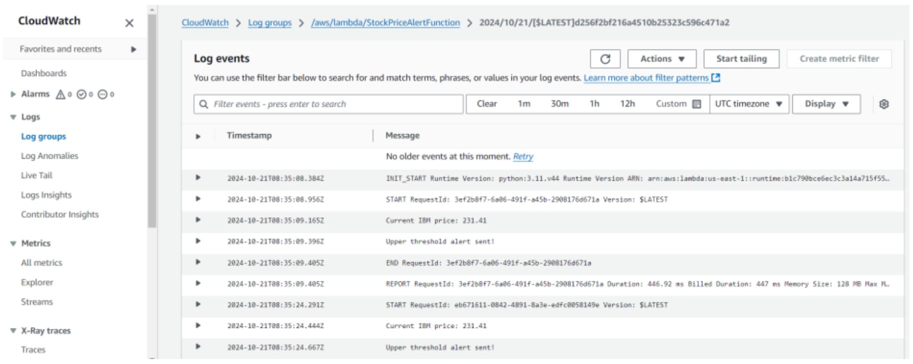


Successfully updated the function StockPriceAlertFunction.

```
▾ Execution results                    Status: Succeeded | Max memory used: 85 MB | Time: 446.92 ms
Test Event Name
Test

Response
{
  "statusCode": 200,
  "body": "\"Checked stock price: $231.41\""
}

Function Logs
START RequestId: 3ef2b8f7-6a06-491f-a45b-2908176d671a Version: $LATEST
Current IBM price: 231.41
Upper threshold alert sent!
END RequestId: 3ef2b8f7-6a06-491f-a45b-2908176d671a
REPORT RequestId: 3ef2b8f7-6a06-491f-a45b-2908176d671a  Duration: 446.92 ms Billed Duration: 447 ms Memory Size: 128 MB Max Memory Used: 85 MB  Init Duration: 570.

Request ID
3ef2b8f7-6a06-491f-a45b-2908176d671a
```

Figure 3.38 Manual Test of Lambda Function

➢ **Monitor Logs:** To track the function’s performance and troubleshoot any issues, you can access detailed logs in CloudWatch Logs. This will show each execution of the Lambda function, including any errors or successful triggers.

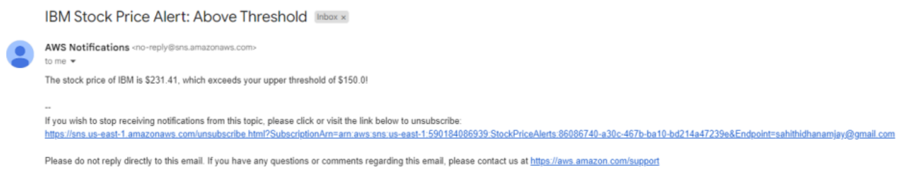


You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

```
Timestamp                   Message
                            No older events at this moment. Retry
2024-10-21T08:35:08.384Z    INIT_START Runtime Version: python:3.11.v44 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:b1c790bce6ec3c3a14a715f55…
2024-10-21T08:35:08.956Z    START RequestId: 3ef2b8f7-6a06-491f-a45b-2908176d671a Version: $LATEST
2024-10-21T08:35:09.165Z    Current IBM price: 231.41
2024-10-21T08:35:09.396Z    Upper threshold alert sent!
2024-10-21T08:35:09.405Z    END RequestId: 3ef2b8f7-6a06-491f-a45b-2908176d671a
2024-10-21T08:35:09.405Z    REPORT RequestId: 3ef2b8f7-6a06-491f-a45b-2908176d671a Duration: 446.92 ms Billed Duration: 447 ms Memory Size: 128 MB Max M…
2024-10-21T08:35:24.291Z    START RequestId: eb671611-0842-4891-8a3e-edfc0058149e Version: $LATEST
2024-10-21T08:35:24.444Z    Current IBM price: 231.41
2024-10-21T08:35:24.667Z    Upper threshold alert sent!
```

Figure 3.39 Monitor the Log Events (CloudWatch)

- **Scheduled Execution:** If the Lambda function is set to run on a schedule via CloudWatch, you can wait for the automatic trigger to execute it at the defined intervals (e.g., every 15 minutes).



IBM Stock Price Alert: Above Threshold

AWS Notifications <no-reply@sns.amazonaws.com>

The stock price of IBM is $231.41, which exceeds your upper threshold of $150.0!

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:590184086939:StockPriceAlerts:86086740-a30c-467b-ba10-bd214a47239e&Endpoint=sahithidhanamjay@gmail.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support

Figure 3.40 Alert Triggered to the Endpoint

CHAPTER 4

CHALLENGES FACED

- **Stock API Rate Limits**:
  - The Alpha Vantage API’s rate limits restricted the number of requests, making it difficult to track multiple stocks simultaneously. Optimizing API calls and caching data helped manage these limits effectively.
- **Threshold Accuracy**:
  - Ensuring that alerts were triggered only for significant stock price changes required fine-tuning the price threshold logic, avoiding unnecessary or missed notifications.
- **Notification Latency**:
  - Ensuring real-time delivery of alerts was crucial, but occasional delays occurred. Monitoring Lambda execution and SNS delivery helped minimize these delays.
- **Scalability**:
  - As the system scaled, managing Lambda execution and API rate limits became critical. Lambda’s serverless model helped handle increased load without manual intervention.
- **Cost Management**:
  - Balancing the costs of Lambda invocations and API requests required careful optimization of the function’s scheduling and efficient API usage to avoid unnecessary expenses.
- **Security and Permissions**:
  - Configuring IAM roles to securely manage service interactions was essential. Implementing least-privilege permissions ensured that each service had only the necessary access.
- **API Failures and Timeouts**:
  - Occasional API failures or timeouts interrupted stock data fetching. Implementing retries and error handling mechanisms ensured that the system remained reliable.
- **Cloud9 Package Installation**:
  - Packaging the requests library for Lambda in Cloud9 required proper zipping of dependencies. Ensuring compatibility post-deployment resolved initial setup challenges.