

INTRODUCTION



INTRODUCTION

- In this project, we will learn how to define and invoke lambda functions using AWS Boto3 SDK.
- Lambda is the most popular and used service in AWS.
- AWS lambda free developers from the worry of provisioning resources, specifying operating systems, managing Hardware, and performing maintenance.
- Simply write your code and run it on Lambda!
 1. Define a Lambda function using Boto3 SDK.
 2. Test the lambda function using Eventbridge (cloudwatch events).
 3. Understand the difference between synchronous and asynchronous invocations.
 4. Invoke a Lambda function using Boto3 SDK.

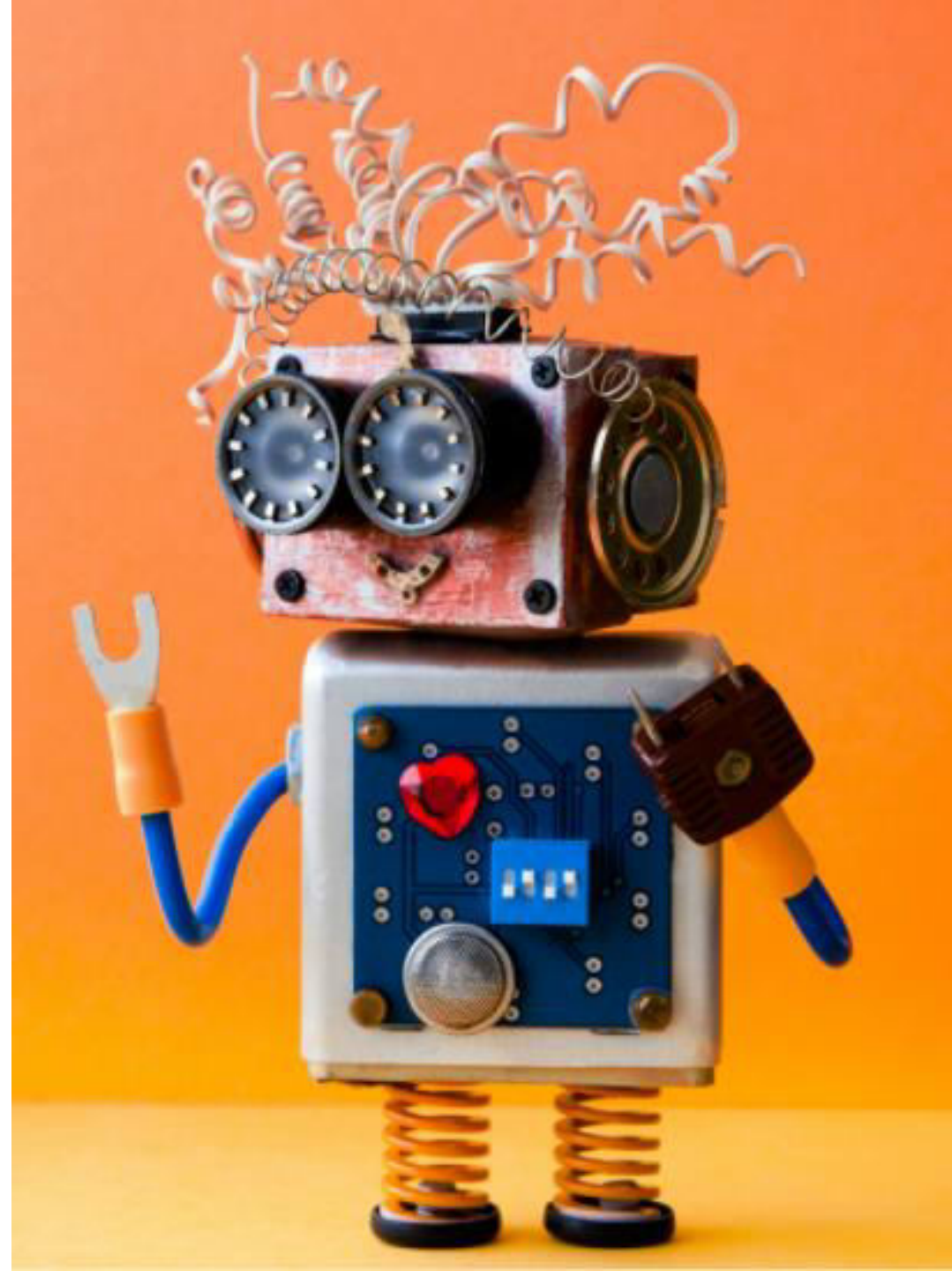


AWS LAMBDA FUNCTION ANATOMY (RECAP)

- **Handler() Function:** Function to be executed upon invocation and it requires two arguments “event” and “context”.
- **Event Object:** data sent during lambda function invocation, for example if a request is made from S3, the event object will contain the bucket key and what kind of action has been performed on the bucket.
- **Context object:** this is generated by the platform and contains information about the underlying infrastructure and execution environment such as allowed runtime and memory.

```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5
6     return {
7         'statusCode': 200,
8         'body': json.dumps('Hello From 50 Days of AWS ML Course!')
9     }
```

SYNCHRONOUS Vs. ASYNCHRONOUS INCOVATIONS



SYNCHRONOUS Vs. ASYNCHRONOUS

- Lambda functions could be invoked (called) using the Console, AWS SDK, Lambda API and AWS Command Line Interface (AWS CLI).
- There are generally two ways of invoking a lambda function: **Synchronously** and **Asynchronously**.
- Lambda functions invocation documentation:
<https://docs.aws.amazon.com/lambda/latest/dg/lambda-invocation.html>

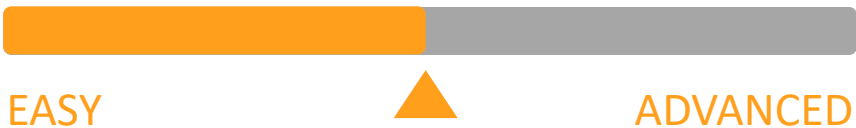
Synchronous Invocation

- With synchronous invocation, you wait for the function to process the event and return a response.
- Synchronous invocations are best suited for Machine Learning workflows.

Asynchronous Invocation

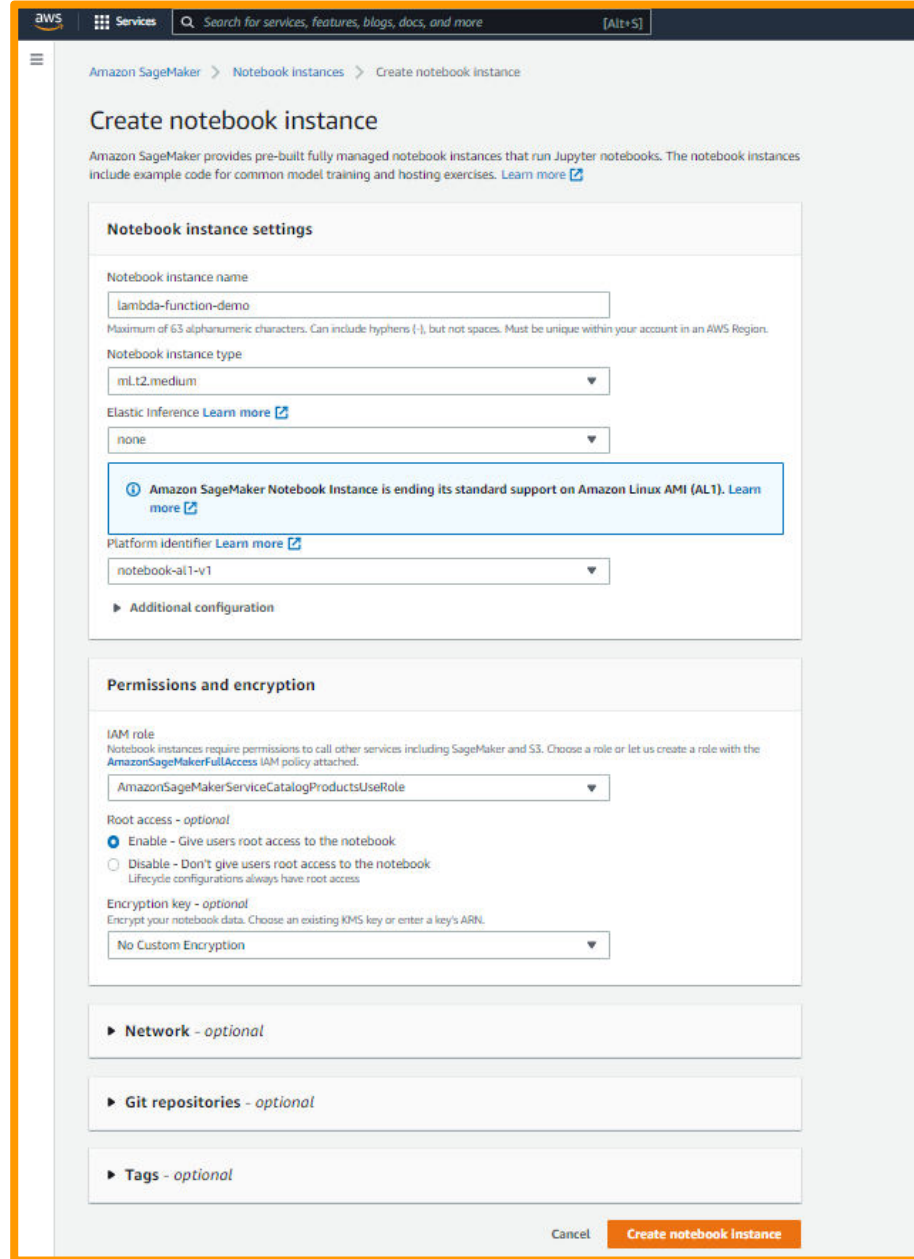
- With asynchronous invocation, Lambda queues the event for processing, so you don't have to wait for a response from Lambda.
- For asynchronous invocation, Lambda handles retries and can send invocation records to a destination.

DEMO: DEFINE AN AWS LAMBDA FUNCTION USING BOTO3 SDK



DEMO: CREATE AN AWS LAMBDA FUNCTION USING BOTO3

CREATE A NOTEBOOK INSTANCE



The screenshot shows the AWS SageMaker console page for creating a notebook instance. The breadcrumb trail is 'Amazon SageMaker > Notebook instances > Create notebook instance'. The page title is 'Create notebook instance'. Below the title, there is a brief description of SageMaker notebook instances and a 'Learn more' link. The main configuration section is titled 'Notebook instance settings' and contains the following fields:

- Notebook instance name:** A text input field with the value 'lambda-function-demo'. A note below the field states: 'Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.'
- Notebook instance type:** A dropdown menu with 'ml.t2.medium' selected.
- Elastic Inference:** A dropdown menu with 'none' selected. A 'Learn more' link is provided.
- Platform identifier:** A dropdown menu with 'notebook-ai1-v1' selected. A 'Learn more' link is provided.

A blue information box with an icon and text states: 'Amazon SageMaker Notebook Instance is ending its standard support on Amazon Linux AMI (AL1). Learn more'. Below the settings section is an 'Additional configuration' section with expandable options:

- Permissions and encryption:**
 - IAM role:** A dropdown menu with 'AmazonSageMakerServiceCatalogProductsUseRole' selected. A note states: 'Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the AmazonSageMakerFullAccess IAM policy attached.'
 - Root access - optional:** Two radio buttons are present: 'Enable - Give users root access to the notebook' (which is selected) and 'Disable - Don't give users root access to the notebook'.
 - Encryption key - optional:** A dropdown menu with 'No Custom Encryption' selected. A note states: 'Encrypt your notebook data. Choose an existing KMS key or enter a key's ARN.'
- Network - optional**
- Git repositories - optional**
- Tags - optional**

At the bottom right of the page, there are two buttons: 'Cancel' and 'Create notebook Instance'.

DEMO: CREATE AN AWS LAMBDA FUNCTION USING BOTO3

CLICK ON OPEN JUPYTER

The screenshot shows the Amazon SageMaker console interface. On the left is a navigation sidebar with sections: Amazon SageMaker, Dashboard, Search, SageMaker Domain (Studio, RStudio, Canvas), Images, and Notebook (Notebook instances, Lifecycle configurations, Git repositories). The main content area is titled 'Amazon SageMaker > Notebook instances'. It features a 'Notebook instances' header with a search bar, a refresh button, an 'Actions' dropdown, and a 'Create notebook instance' button. Below this is a table with columns: Name, Instance, Status, and Actions. A single instance is listed: 'lambda-function-demo' with type 'ml.t2.medium' and status 'InService'. The 'Actions' column for this instance contains the links 'Open Jupyter' and 'Open JupyterLab'. An orange arrow originates from the text 'CLICK ON OPEN JUPYTER' and points directly to the 'Open Jupyter' link.

Name	Instance	Status	Actions
lambda-function-demo	ml.t2.medium	InService	Open Jupyter Open JupyterLab

DEMO: CREATE AN AWS LAMBDA FUNCTION USING BOTO3

GO TO THE SAGEMAKER EXECUTION ROLE AND ATTACH “AWSLAMBDA_FULLACCESS” POLICY

The screenshot shows the AWS IAM console interface. The left sidebar contains navigation links for Identity and Access Management (IAM), including Dashboard, Access management, User groups, Users, Roles, Policies, Identity providers, Account settings, Access reports, Access analyzer, Archive rules, Analyzers, Settings, Credential report, Organization activity, and Service control policies (SCPs). The main content area displays the 'Attach policy to AmazonSageMaker-ExecutionRole-20220121T103405' page. It shows the current permissions policies (2) and a list of other permissions policies (Selected 1/731). A search filter 'lambda' is applied, resulting in 21 matches. The table lists various policies, with 'AWSLambda_FullAccess' selected at the bottom. An orange arrow points from the 'AWSLambda_FullAccess' policy row to the 'Add permissions' button at the top of the console.

Policy name	Type	Description
AWSLambdaBasicExecutionRole-32c9b4c4-34ae-4da3-bafo-77bb9971adff	Customer managed	
AWSLambdaBasicExecutionRole-49e00ba7-3c88-4161-3c1a-5fd2a83c4470	Customer managed	
AWSLambdaBasicExecutionRole-57b18d9d-ce12-4414-912d-80ecd0a24ec4	Customer managed	
AWSLambdaBasicExecutionRole-da0ca0f-9f6f-410d-a771-7ab167d7049	Customer managed	
AWSDeepLensLambdaFunctionAccessPolicy	AWS managed	This policy specifies permissions required by DeepLens Administrative lambda functions that run on a DeepLens device
AWSLambdaDynamoDBExecutionRole	AWS managed	Provides list and read access to DynamoDB streams and write permissions to CloudWatch logs.
AWSCodeDeployRoleForLambda	AWS managed	Provides CodeDeploy service access to perform a Lambda deployment on your behalf.
AWSLambdaExecute	AWS managed	Provides Put, Get access to S3 and full access to CloudWatch Logs.
AWSLambdaSQSQueueExecutionRole	AWS managed	Provides receive message, delete message, and read attribute access to SQS queues, and write permissions to CloudWatch logs.
AWSLambdaKinesisExecutionRole	AWS managed	Provides list and read access to Kinesis streams and write permissions to CloudWatch logs.
AWSLambdaBasicExecutionRole	AWS managed	Provides write permissions to CloudWatch Logs.
AWSLambdaInvocation-DynamoDB	AWS managed	Provides read access to DynamoDB Streams.
AWSLambdaVPCAccessExecutionRole	AWS managed	Provides minimum permissions for a Lambda function to execute while accessing a resource within a VPC - create, describe, delete network interfaces and write permissions to CloudWatch Logs.
AWSLambdaRole	AWS managed	Default policy for AWS Lambda service role.
AWSLambdaENIManagementAccess	AWS managed	Provides minimum permissions for a Lambda function to manage ENIs (create, describe, delete) used by a VPC-enabled Lambda Function.
AWSCodeDeployRoleForLambdaLimited	AWS managed	Provides CodeDeploy service limited access to perform a Lambda deployment on your behalf.
CloudWatchLambdaInsightsExecutionRolePolicy	AWS managed	Policy required for the Lambda Insights Extension
AWSLambdaMSKExecutionRole	AWS managed	Provides permissions required to access MSK Cluster within a VPC, manage ENIs (create, describe, delete) in the VPC and write permissions to CloudWatch Logs.
AWSLambda_ReadOnlyAccess	AWS managed	Grants read-only access to AWS Lambda service, AWS Lambda console features, and other related AWS services.
<input checked="" type="checkbox"/> AWSLambda_FullAccess	AWS managed	Grants full access to AWS Lambda service, AWS Lambda console features, and other related AWS services.

DEMO: CREATE AN AWS LAMBDA FUNCTION USING BOTO3

NOW WE HAVE FULL ACCESS TO LAMBDA!

The screenshot displays the AWS IAM console interface. On the left, the 'Identity and Access Management (IAM)' sidebar is visible, with 'Roles' selected under 'Access management'. The main content area shows the details for the role 'AmazonSageMaker-ExecutionRole-20220121T103405'. The 'Summary' section indicates the role was created on January 21, 2022, and has been active for 26 minutes. The 'Permissions' tab is active, showing three attached policies: 'AmazonSageMaker-ExecutionPolicy-20220121T103405' (Customer managed), 'AmazonSageMakerFullAccess' (AWS managed), and 'AWSLambda_FullAccess' (AWS managed). The 'AWSLambda_FullAccess' policy is highlighted, indicating full access to the AWS Lambda service. The 'Permissions boundary' section is currently set to '(not set)'.

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

- User groups
- Users
- Roles**
- Policies
- Identity providers
- Account settings

Access reports

- Access analyzer
- Archive rules
- Analizers
- Settings
- Credential report
- Organization activity
- Service control policies (SCPs)

AmazonSageMaker-ExecutionRole-20220121T103405

Summary

Creation date
January 21, 2022, 10:34 (UTC-05:00)

Last activity
26 minutes ago

ARN
arn:aws:iam::422132866096:role/service-role/AmazonSageMaker-ExecutionRole-20220121T103405

Maximum session duration
1 hour

Permissions | Trust relationships | Tags | Access Advisor | Revoke sessions

Permissions policies (3)
You can attach up to 10 managed policies.

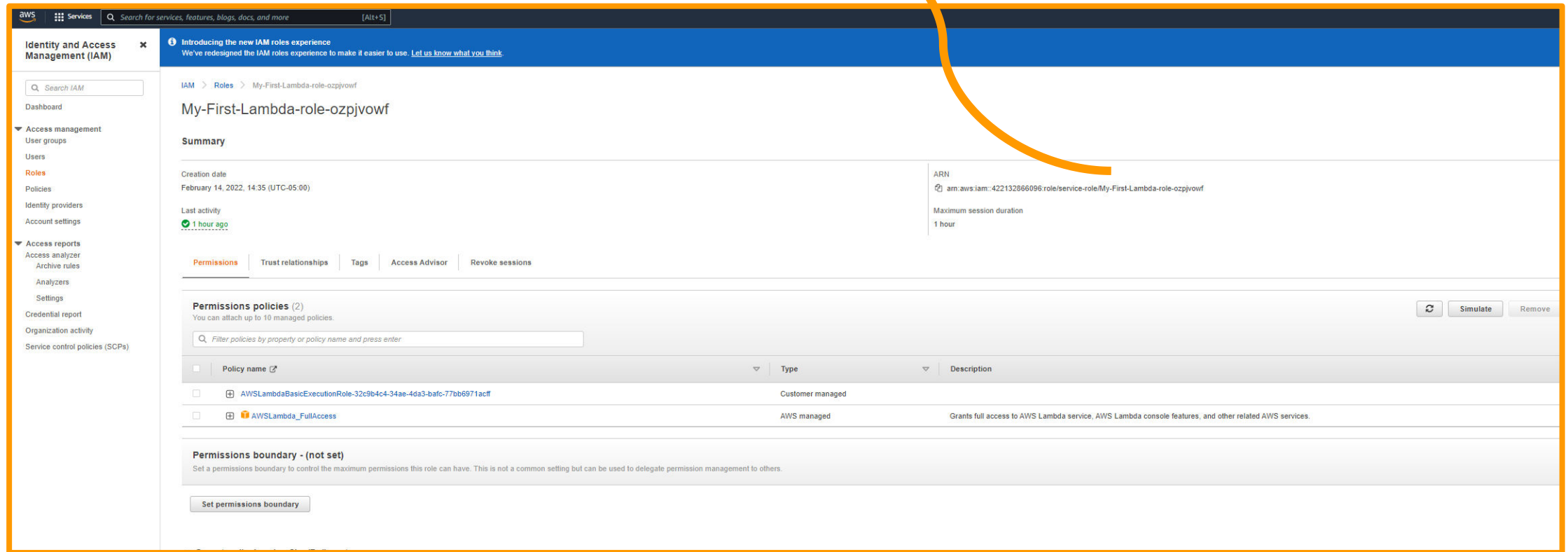
Filter policies by property or policy name and press enter

<input type="checkbox"/>	Policy name	Type	Description
<input type="checkbox"/>	AmazonSageMaker-ExecutionPolicy-20220121T103405	Customer managed	
<input type="checkbox"/>	AmazonSageMakerFullAccess	AWS managed	Provides full access to Amazon SageMaker via the ...
<input type="checkbox"/>	AWSLambda_FullAccess	AWS managed	Grants full access to AWS Lambda service, AWS L...

Permissions boundary - (not set)
Set a permissions boundary to control the maximum permissions this role can have. This is not a common setting but can be used to delegate permission management to others.

DEMO: CREATE AN AWS LAMBDA FUNCTION USING BOTO3

THIS IS THE ROLE ARN YOU WILL NEED IN CODE



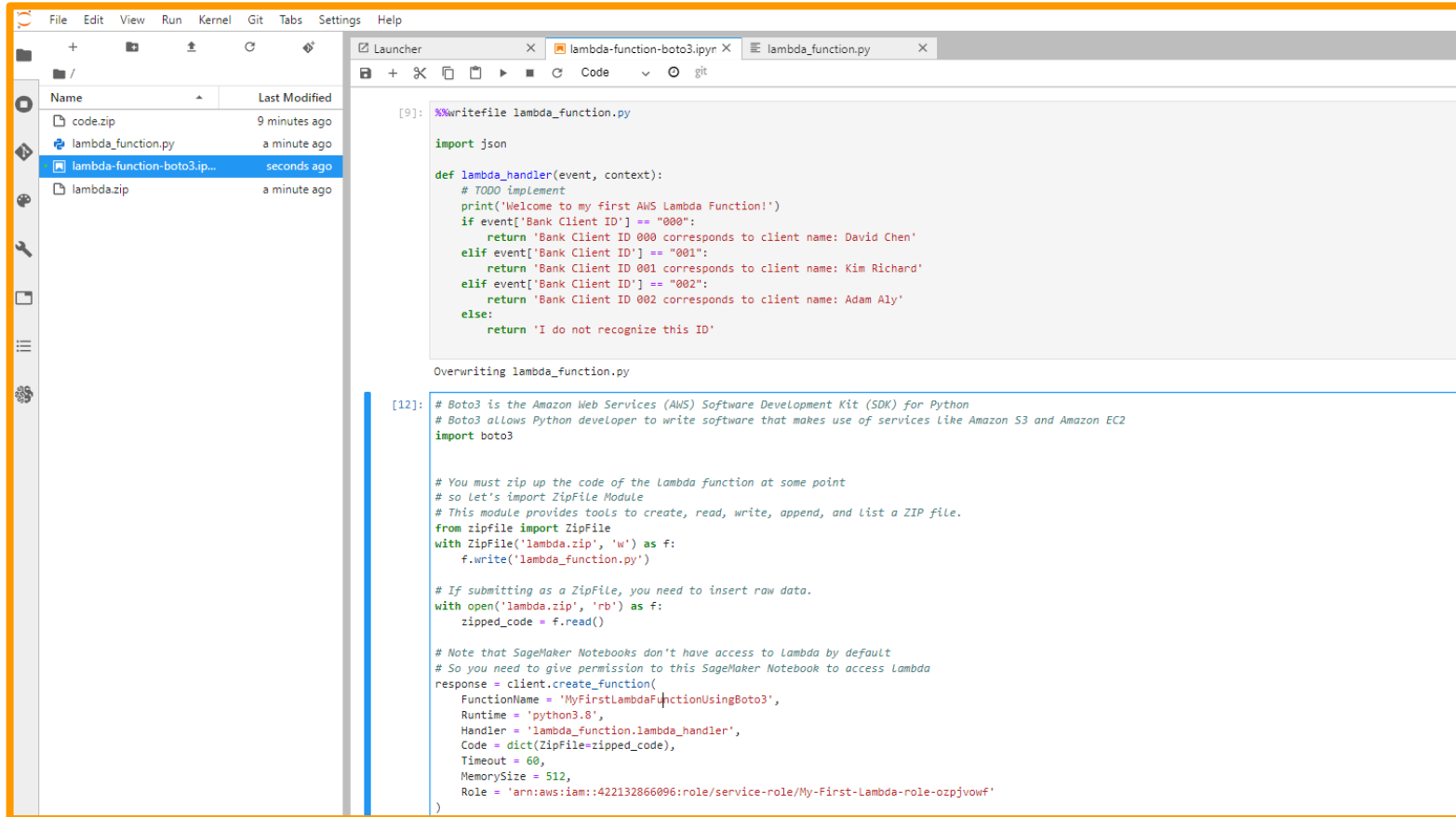
The screenshot displays the AWS IAM console interface. On the left, the 'Identity and Access Management (IAM)' sidebar is visible with navigation links like 'Dashboard', 'Access management', 'Users', 'Roles', 'Policies', etc. The main content area shows the details for a role named 'My-First-Lambda-role-ozpjvowf'. In the 'Summary' section, the 'ARN' is listed as 'arn:aws:iam::422132866096:role/service-role/My-First-Lambda-role-ozpjvowf'. An orange arrow originates from the text 'THIS IS THE ROLE ARN YOU WILL NEED IN CODE' and points directly to this ARN value. Below the summary, there are tabs for 'Permissions', 'Trust relationships', 'Tags', 'Access Advisor', and 'Revoke sessions'. The 'Permissions' tab is active, showing a list of 'Permissions policies (2)'. The first policy is 'AWSLambdaBasicExecutionRole-32c9b4c4-34ae-4da3-bafc-77bb6971acff' (Customer managed), and the second is 'AWSLambda_FullAccess' (AWS managed). At the bottom, there is a section for 'Permissions boundary - (not set)' with a 'Set permissions boundary' button.

Policy name	Type	Description
AWSLambdaBasicExecutionRole-32c9b4c4-34ae-4da3-bafc-77bb6971acff	Customer managed	
AWSLambda_FullAccess	AWS managed	Grants full access to AWS Lambda service, AWS Lambda console features, and other related AWS services.

DEMO: CREATE AN AWS LAMBDA FUNCTION USING BOTO3

RUN THIS CODE, YOU SHOULD SEE A NEWLY CREATED LAMBDA FUNCTION ENTITLED “MyFirstLambdaFunctionUsingBoto3” AVAILABLE.

Note: %%writefile lets you output code developed in a Notebook to a Python module



The screenshot displays a Jupyter Notebook environment with a file explorer on the left and a code editor on the right. The file explorer shows a directory with files: code.zip, lambda_function.py, lambda-function-boto3.ipynb (selected), and lambda.zip. The code editor has two tabs: 'lambda-function-boto3.ipynb' and 'lambda_function.py'. The active cell in the notebook is a code cell with the following content:

```
[9]: %%writefile lambda_function.py

import json

def lambda_handler(event, context):
    # TODO implement
    print('Welcome to my first AWS Lambda Function!')
    if event['Bank Client ID'] == "000":
        return 'Bank Client ID 000 corresponds to client name: David Chen'
    elif event['Bank Client ID'] == "001":
        return 'Bank Client ID 001 corresponds to client name: Kim Richard'
    elif event['Bank Client ID'] == "002":
        return 'Bank Client ID 002 corresponds to client name: Adam Aly'
    else:
        return 'I do not recognize this ID'
```

Below the code cell, a message indicates 'Overwriting lambda_function.py'. The next cell is a code cell with the following content:

```
[12]: # Boto3 is the Amazon Web Services (AWS) Software Development Kit (SDK) for Python
# Boto3 allows Python developer to write software that makes use of services like Amazon S3 and Amazon EC2
import boto3

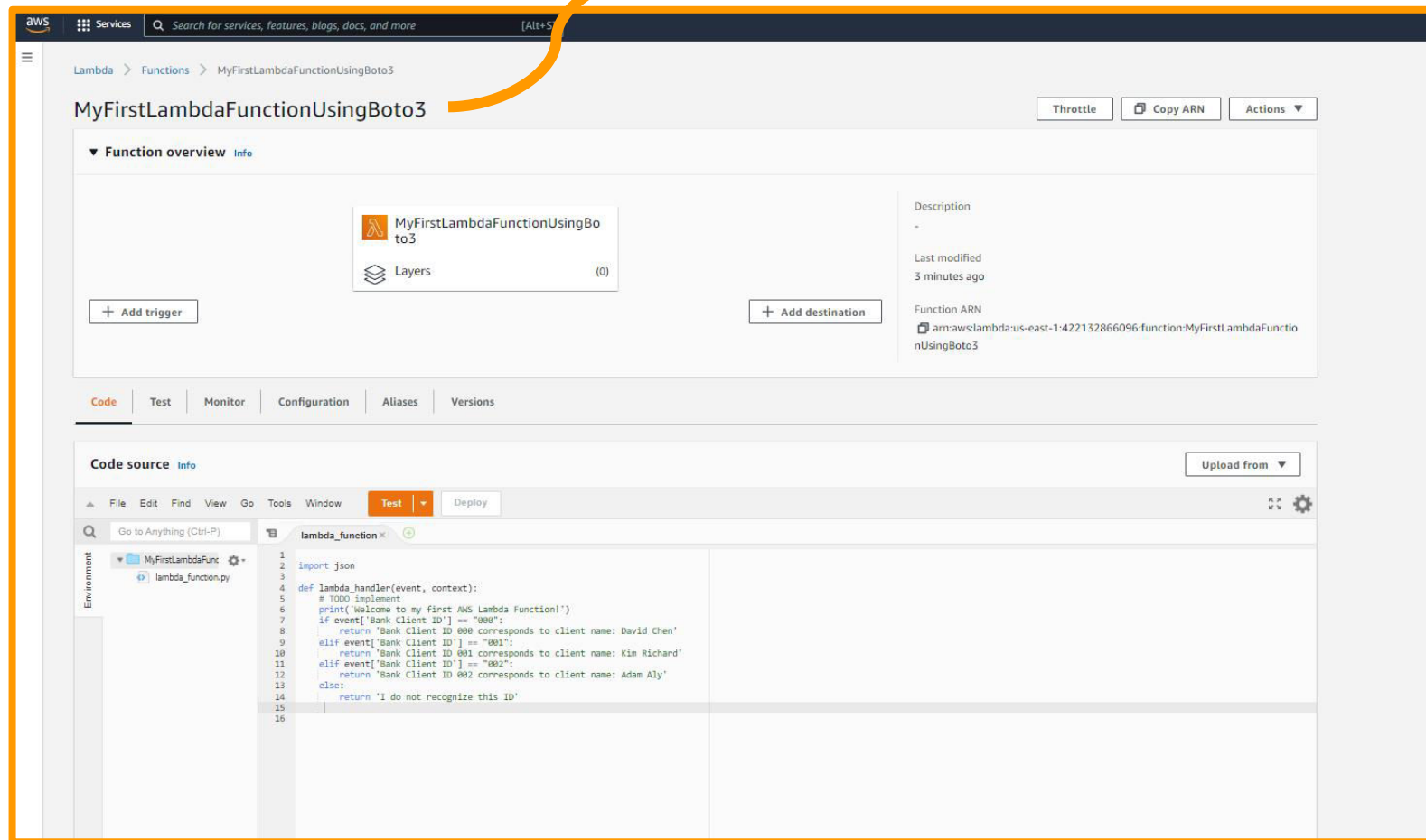
# You must zip up the code of the Lambda function at some point
# so let's import ZipFile Module
# This module provides tools to create, read, write, append, and list a ZIP file.
from zipfile import ZipFile
with ZipFile('lambda.zip', 'w') as f:
    f.write('lambda_function.py')

# If submitting as a ZipFile, you need to insert raw data.
with open('lambda.zip', 'rb') as f:
    zipped_code = f.read()

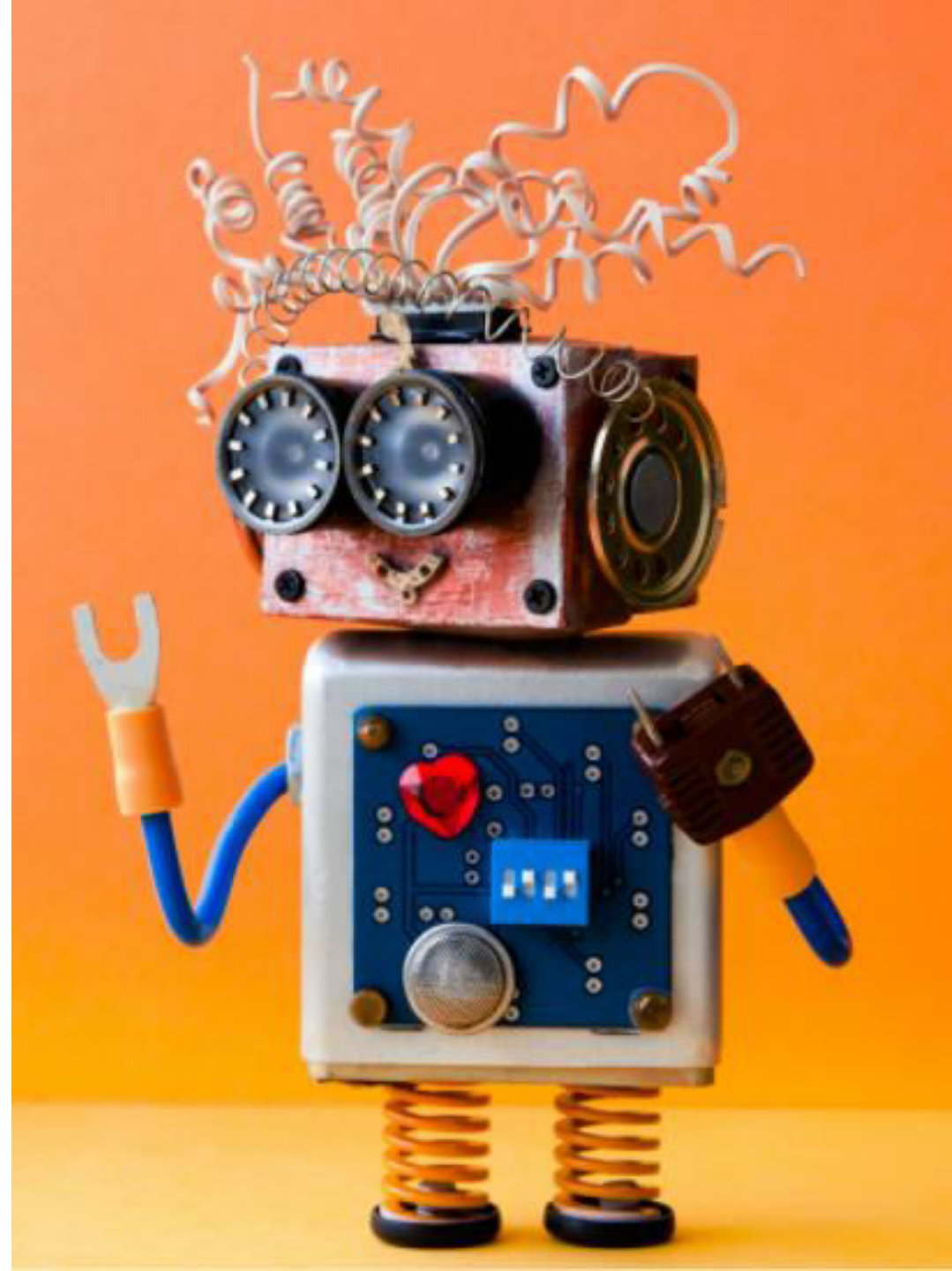
# Note that SageMaker Notebooks don't have access to Lambda by default
# So you need to give permission to this SageMaker Notebook to access Lambda
response = client.create_function(
    FunctionName = 'MyFirstLambdaFunctionUsingBoto3',
    Runtime = 'python3.8',
    Handler = 'lambda_function.lambda_handler',
    Code = dict(ZipFile=zipped_code),
    Timeout = 60,
    MemorySize = 512,
    Role = 'arn:aws:iam::422132866096:role/service-role/My-First-Lambda-role-ozpjvowf'
)
```

DEMO: CREATE AN AWS LAMBDA FUNCTION USING BOTO3

A NEW LAMBDA FUNCTION ENTITLED
“MyFirstLambdaFunctionUsingBoto3” HAS BEEN CREATED USING
BOTO3 SDK



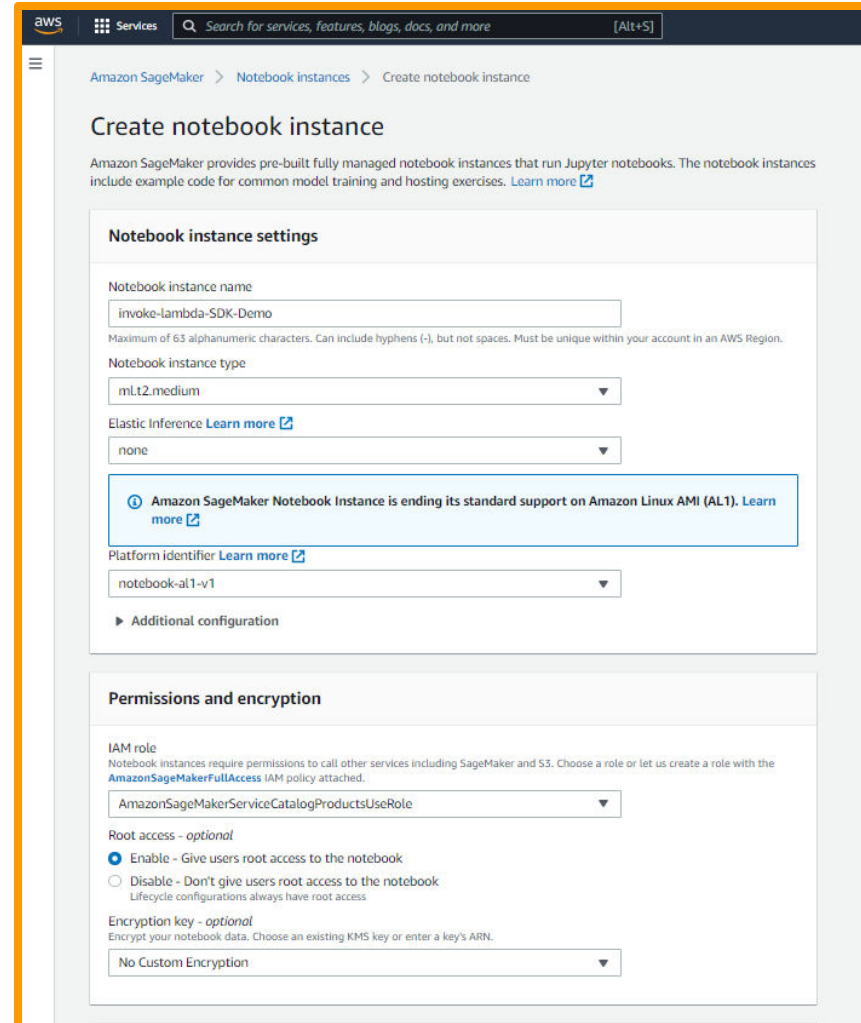
DEMO: LAMBDA INVOCATION WITH BOTO3 SDK



DEMO: LAMBDA INVOCATION WITH SDK

- Great Boto3 Documentation:
<https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/lambda.html#Lambda.Client.invoke>

GO TO SAGEMAKER AND
START A NEW NOTEBOOK



The screenshot shows the AWS SageMaker console page for creating a new notebook instance. The page is titled 'Create notebook instance' and includes a breadcrumb trail: 'Amazon SageMaker > Notebook instances > Create notebook instance'. Below the title, there is a brief description of SageMaker notebook instances and a 'Learn more' link. The main section is 'Notebook instance settings', which contains several fields: 'Notebook instance name' (set to 'invoke-lambda-SDK-Demo'), 'Notebook instance type' (set to 'ml.t2.medium'), 'Elastic Inference' (set to 'none'), and 'Platform identifier' (set to 'notebook-ml1-v1'). A blue warning box is present, stating: 'Amazon SageMaker Notebook Instance is ending its standard support on Amazon Linux AMI (AL1). Learn more'. Below the settings section is a collapsed 'Additional configuration' section. The 'Permissions and encryption' section is also visible, showing the 'IAM role' set to 'AmazonSageMakerServiceCatalogProductsUseRole' and 'Root access' set to 'Enable - Give users root access to the notebook'. The 'Encryption key' is set to 'No Custom Encryption'.

aws Services Search for services, features, blogs, docs, and more [Alt+S]

Amazon SageMaker > Notebook instances > Create notebook instance

Create notebook instance

Amazon SageMaker provides pre-built fully managed notebook instances that run Jupyter notebooks. The notebook instances include example code for common model training and hosting exercises. [Learn more](#)

Notebook instance settings

Notebook instance name
invoke-lambda-SDK-Demo
Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

Notebook instance type
ml.t2.medium

Elastic Inference [Learn more](#)
none

Amazon SageMaker Notebook Instance is ending its standard support on Amazon Linux AMI (AL1). [Learn more](#)

Platform identifier [Learn more](#)
notebook-ml1-v1

► Additional configuration

Permissions and encryption

IAM role
Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the [AmazonSageMakerFullAccess](#) IAM policy attached.
AmazonSageMakerServiceCatalogProductsUseRole

Root access - optional
☒ Enable - Give users root access to the notebook
☐ Disable - Don't give users root access to the notebook
Lifecycle configurations always have root access

Encryption key - optional
Encrypt your notebook data. Choose an existing KMS key or enter a key's ARN.
No Custom Encryption

DEMO: LAMBDA INVOCATION WITH SDK

LET'S DEFINE A NEW FUNCTION AND THEN INVOKE IT IN THE SDK

TASK #1. DEFINE A LAMBDA FUNCTION

```
In [ ]: %%writefile lambda_function.py
```

```
import json

def lambda_handler(event, context):
    # TODO implement
    print('Welcome to my first AWS Lambda Function!')
    if event['Bank Client ID'] == "000":
        print('Bank Client ID 000 corresponds to client name: David Chen')
    elif event['Bank Client ID'] == "001":
        print('Bank Client ID 001 corresponds to client name: Kim Richard')
    elif event['Bank Client ID'] == "002":
        print('Bank Client ID 002 corresponds to client name: Adam Aly')
    else:
        return 'I do not recognize this ID'
```

```
In [ ]: # Boto3 is the Amazon Web Services (AWS) Software Development Kit (SDK) for Python
# Boto3 allows Python developer to write software that makes use of services like Amazon S3 and Amazon EC2
import boto3
```

```
# You must zip up the code of the Lambda function at some point
# so let's import ZipFile Module
# This module provides tools to create, read, write, append, and list a ZIP file.
from zipfile import ZipFile
with ZipFile('lambda.zip', 'w') as f:
    f.write('lambda_function.py')

# If submitting as a ZipFile, you need to insert raw data.
with open('lambda.zip', 'rb') as f:
    zipped_code = f.read()

# Note that SageMaker Notebooks don't have access to Lambda by default
# So you need to give permission to this SageMaker Notebook to access Lambda
response = client.create_function(
    FunctionName = 'MyThirdLambdaFunctionUsingBoto3',
    Runtime = 'python3.8',
    Handler = 'lambda_function.lambda_handler',
    Code = dict(ZipFile=zipped_code),
    Timeout = 60,
    MemorySize = 512,
    Role = 'arn:aws:iam::422132866096:role/service-role/My-First-Lambda-role-ozpjvowf'
)
```

TASK #2: LET'S INVOKE THIS LAMBDA FUNCTION USING SDK

```
In [ ]: import boto3
import json
client = boto3.client('lambda')
```

```
In [ ]: # FunctionName: write a function name
# InvocationType (string)
# 1. RequestResponse (default) - Invoke the function synchronously. Keep the connection open until the function returns a response
# 2. Event - Invoke the function asynchronously. Send events that fail multiple times to the function's dead-letter queue (if configured)
# 3. DryRun - Validate parameter values and verify that the user or role has permission to invoke the function.

# LogType (string): Set to Tail to include the execution log in the response. Applies to synchronously invoked functions only
# ClientContext (string): Up to 3583 bytes of base64-encoded data about the invoking client to pass to the function in the context
# Payload (bytes or seekable file-like object): The JSON that you want to provide to your Lambda function as input.
# You can enter the JSON directly. For example, --payload '{ "key": "value" }'
# You can also specify a file path. For example, --payload file://payload.json
```

```
response = client.invoke(
    FunctionName = 'MyThirdLambdaFunctionUsingBoto3',
    InvocationType = 'Event',
    LogType = 'Tail',
    ClientContext = 'string',
    Payload = json.dumps({'Bank Client ID': '003'}).encode('utf-8'),
)
```

```
In [ ]: print(response)
```

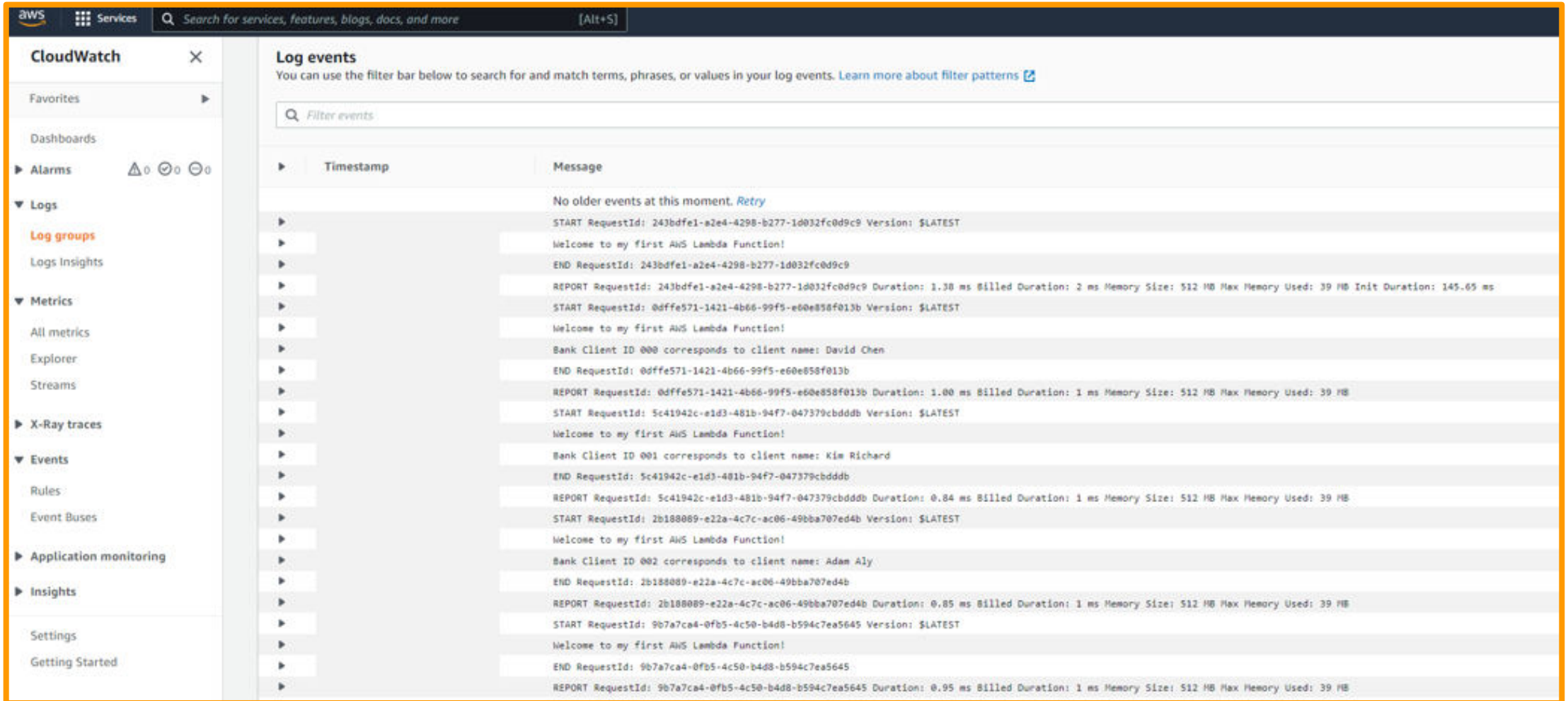
```
In [ ]:
```

DEMO: LAMBDA INVOCATION WITH SDK

- **FunctionName:** write a function name
- **InvocationType (string)**
 1. RequestResponse (default): Invoke the function synchronously.
 2. Event: Invoke the function asynchronously.
 3. DryRun: Validate parameter values and verify that the user or role has permission to invoke the function.
- **LogType (string):** Set to Tail to include the execution log in the response. Applies to synchronously invoked functions only.
- **ClientContext (string):** Up to 3583 bytes of base64-encoded data about the invoking client to pass to the function in the context object.
- **Payload:** The JSON that you want to provide to your Lambda function as input.
- You can enter the JSON directly. For example, `--payload '{ "key": "value" }'`
- You can also specify a file path. For example, `--payload file://payload.json`

DEMO: LAMBDA INVOCATION WITH SDK

GO TO CLOUDWATCH AND EXPLORE THE LOGS



The screenshot displays the AWS CloudWatch console interface. On the left, a navigation sidebar includes links to CloudWatch, Favorites, Dashboards, Alarms, Logs, Metrics, X-Ray traces, Events, Application monitoring, and Insights. The 'Logs' section is expanded, showing 'Log groups' and 'Logs Insights'. The main panel, titled 'Log events', contains a search bar and a table of log entries. The table has two columns: 'Timestamp' and 'Message'. The log entries show a sequence of events for a Lambda function, including 'START', 'Welcome to my first AWS Lambda Function!', 'END', and 'REPORT' messages, each with associated metadata like RequestId, Duration, and Memory Size.

Timestamp	Message
	No older events at this moment. Retry
	START RequestId: 243bdfef-a2e4-4298-b277-1d032fc0d9c9 Version: \$LATEST
	Welcome to my first AWS Lambda Function!
	END RequestId: 243bdfef-a2e4-4298-b277-1d032fc0d9c9
	REPORT RequestId: 243bdfef-a2e4-4298-b277-1d032fc0d9c9 Duration: 1.38 ms Billed Duration: 2 ms Memory Size: 512 MB Max Memory Used: 39 MB Init Duration: 145.65 ms
	START RequestId: 0dffe571-1421-4b66-99f5-e60e858f013b Version: \$LATEST
	Welcome to my first AWS Lambda Function!
	Bank Client ID 000 corresponds to client name: David Chen
	END RequestId: 0dffe571-1421-4b66-99f5-e60e858f013b
	REPORT RequestId: 0dffe571-1421-4b66-99f5-e60e858f013b Duration: 1.00 ms Billed Duration: 1 ms Memory Size: 512 MB Max Memory Used: 39 MB
	START RequestId: 5c41942c-e1d3-481b-94f7-047379cbdddb Version: \$LATEST
	Welcome to my first AWS Lambda Function!
	Bank Client ID 001 corresponds to client name: Kim Richard
	END RequestId: 5c41942c-e1d3-481b-94f7-047379cbdddb
	REPORT RequestId: 5c41942c-e1d3-481b-94f7-047379cbdddb Duration: 0.84 ms Billed Duration: 1 ms Memory Size: 512 MB Max Memory Used: 39 MB
	START RequestId: 2b188089-e22a-4c7c-ac06-49bba707ed4b Version: \$LATEST
	Welcome to my first AWS Lambda Function!
	Bank Client ID 002 corresponds to client name: Adam Aly
	END RequestId: 2b188089-e22a-4c7c-ac06-49bba707ed4b
	REPORT RequestId: 2b188089-e22a-4c7c-ac06-49bba707ed4b Duration: 0.85 ms Billed Duration: 1 ms Memory Size: 512 MB Max Memory Used: 39 MB
	START RequestId: 9b7a7ca4-0fb5-4c50-b4d8-b594c7ea5645 Version: \$LATEST
	Welcome to my first AWS Lambda Function!
	END RequestId: 9b7a7ca4-0fb5-4c50-b4d8-b594c7ea5645
	REPORT RequestId: 9b7a7ca4-0fb5-4c50-b4d8-b594c7ea5645 Duration: 0.95 ms Billed Duration: 1 ms Memory Size: 512 MB Max Memory Used: 39 MB

DEMO: LAMBDA INVOCATION WITH SDK

NOTE THAT YOU MIGHT NEED TO ATTACH CLOUDWATCH FULL ACCESS POLICY TO YOUR LAMBDA FUNCTION ROLE TO SEE THE LOGS

The screenshot shows the AWS IAM console interface. The top navigation bar includes the S3 and Amazon SageMaker logos. The left sidebar contains the 'Identity and Access Management (IAM)' section with a search bar and a list of navigation items: Dashboard, Access management (expanded), Roles (highlighted), Policies, Identity providers, Account settings, Access reports (expanded), Access analyzer, Archive rules, Analyzers, Settings, Credential report, Organization activity, and Service control policies (SCPs). The main content area displays the details for the role 'my-second-lambda-function-role-apraph6n'. The 'Summary' tab is active, showing the role's creation date (May 01, 2022, 13:26 UTC-04:00), last activity (8 hours ago), ARN (arn:aws:iam::971421653261:role/service-role/my-second-lambda-function-role-apraph6n), and maximum session duration (1 hour). Below the summary, there are tabs for Permissions, Trust relationships, Tags, Access Advisor, and Revoke sessions. The 'Permissions' tab is selected, showing 'Permissions policies (1)'. A table lists the attached policy: 'AWSLambdaBasicExecutionRole-8cecf5d-019d-4967-a78d-9b...' with a 'Customer managed' type. The table has columns for Policy name, Type, and Description. At the top right of the permissions section, there are buttons for 'Delete', 'Edit', 'Simulate', 'Remove', and 'Add permissions'.

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

- User groups
- Users
- Roles**
- Policies
- Identity providers
- Account settings

Access reports

- Access analyzer
- Archive rules
- Analyzers
- Settings
- Credential report
- Organization activity
- Service control policies (SCPs)

my-second-lambda-function-role-apraph6n

Summary

Creation date
May 01, 2022, 13:26 (UTC-04:00)

Last activity
8 hours ago

ARN
arn:aws:iam::971421653261:role/service-role/my-second-lambda-function-role-apraph6n

Maximum session duration
1 hour

Permissions Trust relationships Tags Access Advisor Revoke sessions

Permissions policies (1)
You can attach up to 10 managed policies.

Filter policies by property or policy name and press enter

<input type="checkbox"/>	Policy name	Type	Description
<input type="checkbox"/>	AWSLambdaBasicExecutionRole-8cecf5d-019d-4967-a78d-9b...	Customer managed	

DEMO: LAMBDA INVOCATION WITH EVENTBRIDGE



DEMO: LAMBDA INVOCATION WITH EVENTBRIDGE

CREATE A BASIC LAMBDA FUNCTION NAMED “basic-lambda”

The screenshot displays the AWS Lambda console interface for a function named 'basic-lambda'. The top navigation bar includes the AWS logo, 'Services', and a search bar. The breadcrumb trail shows 'Lambda > Functions > basic-lambda'. The function name 'basic-lambda' is prominently displayed at the top left of the main content area, with buttons for 'Throttle', 'Copy ARN', and 'Actions' to its right. Below this, the 'Function overview' section is expanded, showing a diagram of the function's architecture. It includes a box for 'basic-lambda' with a 'Layers' section indicating '(0)' layers. Below this is a box for 'EventBridge (CloudWatch Events)' with an '+ Add trigger' button. To the right of the diagram is a '+ Add destination' button. Further right, a 'Description' field is empty, and 'Last modified' is listed as '9 minutes ago'. The 'Function ARN' is shown as 'arn:aws:lambda:us-east-1:422132866096:function:basic-lambda'. Below the overview, a horizontal tab bar contains 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Code' tab is selected, showing the 'Code source' section with an 'Upload from' dropdown. Below this is a code editor with a menu bar (File, Edit, Find, View, Go, Tools, Window) and buttons for 'Test' and 'Deploy'. The code editor shows a file named 'lambda_function.py' with the following Python code:

```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from 50 Days of AWS ML Course!')}
8
9
```

The left sidebar of the code editor shows the 'Environment' pane with a tree view containing 'basic-lambda' and 'lambda_function.py'.

DEMO: LAMBDA INVOCATION WITH EVENTBRIDGE

GO TO CLOUDWATCH AND CLICK RULES

The screenshot shows the AWS CloudWatch console interface. The left sidebar contains navigation links: CloudWatch, Favorites, Dashboards, Alarms (with 0 alerts), Logs, Metrics, X-Ray traces, Events (expanded), Rules, Event Buses, Application monitoring, and Insights. The main content area displays the 'Alarms by AWS service' page. At the top, there are two informational banners: one about 'Introducing Amazon CloudWatch Metrics Insights' and another about the 'new design for CloudWatch Automatic Dashboards'. Below these, the 'CloudWatch: Overview' section shows a dropdown for 'All resources'. The 'Alarms by AWS service' section features a table with columns for 'Status', 'Alarm', 'Insufficient', and 'OK'. The table lists several services: API Gateway, CloudWatch Logs, EFS, Lambda, S3, Step Functions, and Usage, all with a status of '-'. To the right of the table, the 'Recent alarms' section is empty, displaying a message: 'Recent alarms will appear here. Learn more about CloudWatch Alarms.' The top navigation bar includes the AWS logo, 'Services' link, a search bar, and a '[Alt+S]' shortcut.

CloudWatch: Overview

All resources

Alarms by AWS service

Status	Alarm	Insufficient	OK
API Gateway	-	-	-
CloudWatch Logs	-	-	-
EFS	-	-	-
Lambda	-	-	-
S3	-	-	-
Step Functions	-	-	-
Usage	-	-	-

Recent alarms

Recent alarms will appear here.
[Learn more about CloudWatch Alarms.](#)

DEMO: LAMBDA INVOCATION WITH EVENTBRIDGE

CLOUDWATCH EVENT IS NOW CALLED EVENTBRIDGE,
CLICK ON GO TO AMAZON EVENTBRIDGE

The screenshot shows the AWS Management Console interface. At the top, there's a navigation bar with the AWS logo, a 'Services' menu, and a search bar. Below the navigation bar, a sidebar on the left contains a 'CloudWatch' section with a close button (X). The sidebar lists various services: Favorites, Dashboards, Alarms, Logs, Metrics, X-Ray traces, Events (with a sub-item 'Rules' highlighted), Application monitoring, and Insights. The main content area features a dark blue header with the title 'CloudWatch Events is now EventBridge'. Below the title, a paragraph explains that Amazon EventBridge builds upon and extends CloudWatch Events, maintaining the same API and endpoint. Three buttons are provided: 'Go to Amazon EventBridge' (orange), 'Go to EventBridge documentation' (white with a link icon), and 'Back to CloudWatch Events' (white). The main content area also lists 'Additional capabilities in Amazon EventBridge' with three columns: 'Integrated SaaS partner event sources', 'Schema registries', 'API destinations', 'Custom and partner event buses', and 'Archive and replay events'.

CloudWatch Events is now EventBridge

Amazon EventBridge builds upon and extends CloudWatch Events. It uses the same service API and endpoint, and the same underlying service infrastructure. For existing CloudWatch Events customers, nothing changes - you can continue to use the same API, and CloudFormation templates. All existing same way in EventBridge. Existing default event bus, rules and events can also be accessed in the Amazon EventBridge console.

[Go to Amazon EventBridge](#) [Go to EventBridge documentation](#) [Back to CloudWatch Events](#)

Additional capabilities in Amazon EventBridge

Integrated SaaS partner event sources

Receive events from SaaS partner applications and services such as Zendesk, MongoDB, and PagerDuty. You don't need to manage any integration setup such as authentication. Events from your SaaS provider simply appear on your event bus.

Schema registries

Automatically discover schemas of events on your event buses, and store them in schema registries. Add your own custom schemas and registries. Schemas are searchable and accessible by developers across your organization. Generate code-bindings, such as Java, Python or TypeScript in your IDE for any event schemas.

API destinations

Use API destinations to send events to any public HTTP API - hosted in AWS, on-premises, your custom applications, or in a SaaS partner, with the ability to control throughput and authorization. No custom code or additional infrastructure required.

Custom and partner event buses

EventBridge supports default, custom and partner event buses. The default even bus receives events from Amazon services. A custom event bus sends events to or receives events from a different account/Region. A partner event bus receives events from a SaaS partner.

Archive and replay events

Replay archived events to reprocess past events back to an event bus or a specific EventBridge rule. This feature allows you to recover from errors by hydrating targets with historic events, and recover from errors.

DEMO: LAMBDA INVOCATION WITH EVENTBRIDGE

CLICK ON CREATE RULE

The screenshot shows the Amazon EventBridge console interface. On the left is a navigation sidebar with links to 'Getting started', 'Event buses', 'Rules' (highlighted), 'Archives', 'Replays', 'Integration', 'Partner event sources', 'API destinations', 'Schema registry', and 'Schemas'. Below these is a 'Documentation' link. The main content area has a blue header with 'EventBridge - Learning content' and a 'Provide Feedback' button. Below the header, the breadcrumb 'Amazon EventBridge > Rules' is shown. The title 'Rules' is followed by a description: '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.' Below this is a 'Select event bus' section with a dropdown menu currently set to 'default'. At the bottom, there's a 'Rules (0/0)' section with a search bar, a status filter set to 'Any status', and a row of action buttons: 'Refresh', 'Edit', 'Delete', 'Enable', and 'Create rule' (which is orange). Below the buttons is a table with columns 'Name', 'Status', 'Type', and 'Description'. The table is empty, showing 'No rules' and 'No rules to display', with a 'Create rule' button at the bottom.

DEMO: LAMBDA INVOCATION WITH EVENTBRIDGE

LET'S INVOKE THE LAMBDA FUNCTION ONCE EVERY 2 MINUTES

Amazon EventBridge

Getting started

Event buses

Rules

Archives

Replays

Integration

Partner event sources

API destinations

Schema registry

Schemas

Documentation

EventBridge - Learning content

Tell us what topics you would like to see more learning material for (tutorials, videos, blog posts, etc.).

Amazon EventBridge > Rules > Create rule

Create rule

A rule watches for certain events and then routes them to AWS targets that you choose. You can create a rule that performs an AWS action automatically when another AWS action happens, or a rule that performs an AWS action regularly on a set schedule.

Name and description

Name

invoke-lambda-demo

Maximum of 64 characters consisting of lower/upper case letters, -, ., and _.

Description - optional

Enter description

Define pattern

Build or customize an Event Pattern or set a Schedule to invoke Targets.

Event pattern

Build a pattern to match events

Schedule

Invoke your targets on a schedule

Fixed rate every

2

Minutes

Cron expression

CRON expression have six required fields, which are separated by white space. [Learn more about CRON expression.](#) Enter CRON expression below to see the next 10 trigger date(s).

0/5 * * * * *

Sample event(s)

Select targets

Select target(s) to invoke when an event matches your event pattern or when schedule is triggered (limit of 5 targets per rule).

Target

Select target(s) to invoke when an event matches your event pattern or when schedule is triggered (limit of 5 targets per rule).

Lambda function

Function

basic-lambda

Configure version/alias

Configure input

Retry policy and dead-letter queue

Add target

Tags - optional

No tags associated with the resource.

Add tag

Cancel

Update

DEMO: LAMBDA INVOCATION WITH EVENTBRIDGE

NOW THE RULE HAS BEEN ACTIVATED, LET'S SEE IF IT WORKS!

The screenshot displays the AWS EventBridge console interface. At the top, there's a navigation bar with the AWS logo, 'Services' link, a search bar, and user information (N. Virginia, Ryan). A left-hand navigation pane lists various EventBridge features like 'Getting started', 'Event buses', 'Rules' (highlighted), 'Archives', 'Replays', 'Integration', 'Partner event sources', 'API destinations', 'Schema registry', 'Schemas', and 'Documentation'. The main content area shows a 'Rules' page with a breadcrumb 'Amazon EventBridge > Rules'. Below this, a message states 'Rule invoke-lambda-demo was created successfully'. The 'Rules' section includes a description: '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.' There's a 'Select event bus' section with a dropdown menu currently set to 'default'. Below this is a table of rules. The table has columns for 'Name', 'Status', 'Type', and 'Description'. One rule is listed: 'invoke-lambda-demo' with a status of 'Enabled' (indicated by a green checkmark) and a type of 'Scheduled Standard'. Action buttons like 'Refresh', 'Edit', 'Delete', 'Enable', and 'Create rule' are visible at the top right of the rules list.

Amazon EventBridge - Learning content
Tell us what topics you would like to see more learning material for (tutorials, videos, blog posts, etc.).

Provide Feedback

Rule invoke-lambda-demo was created successfully

Amazon EventBridge > Rules

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

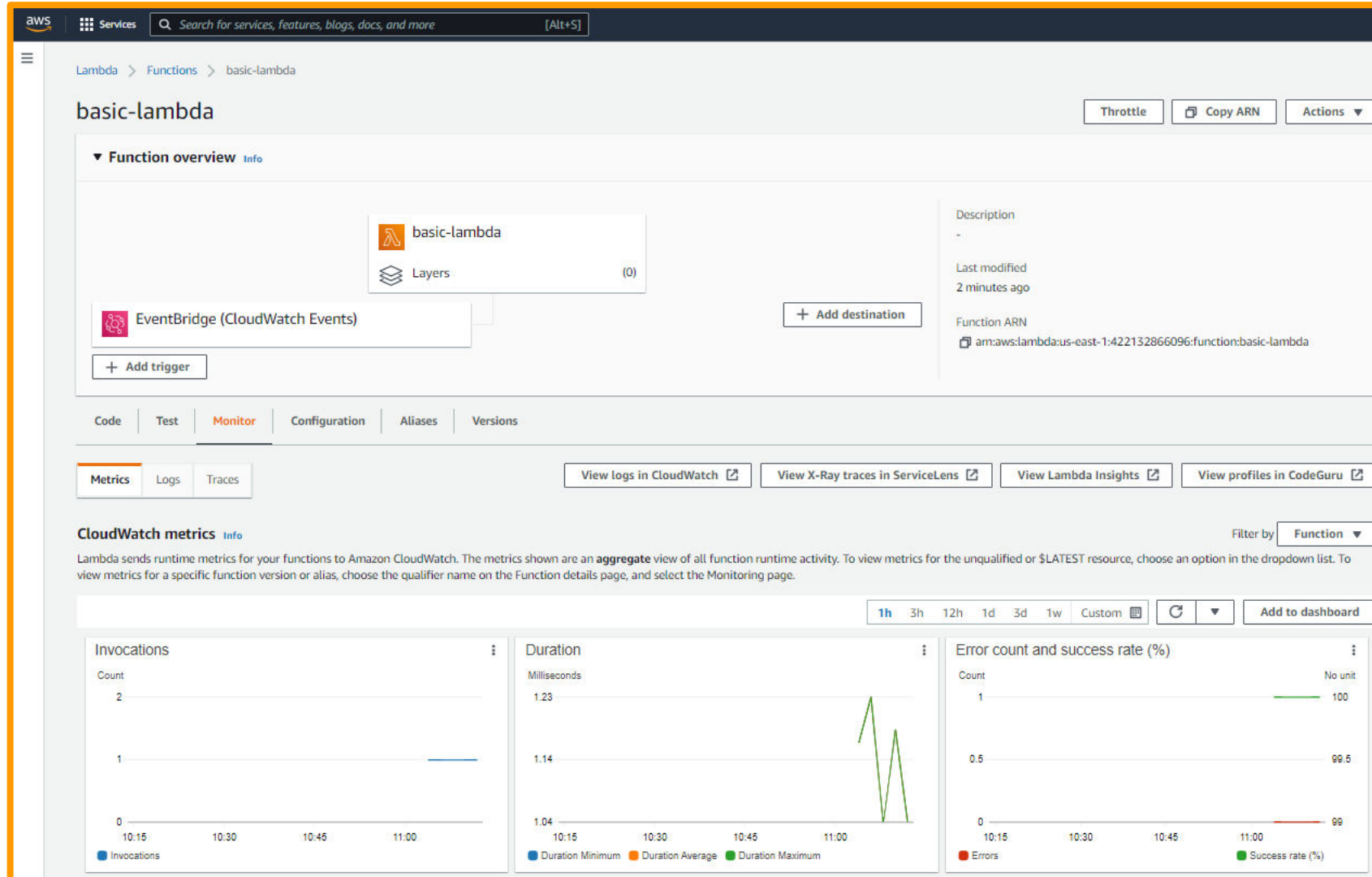
Rules (1/1)

Find rules Any status

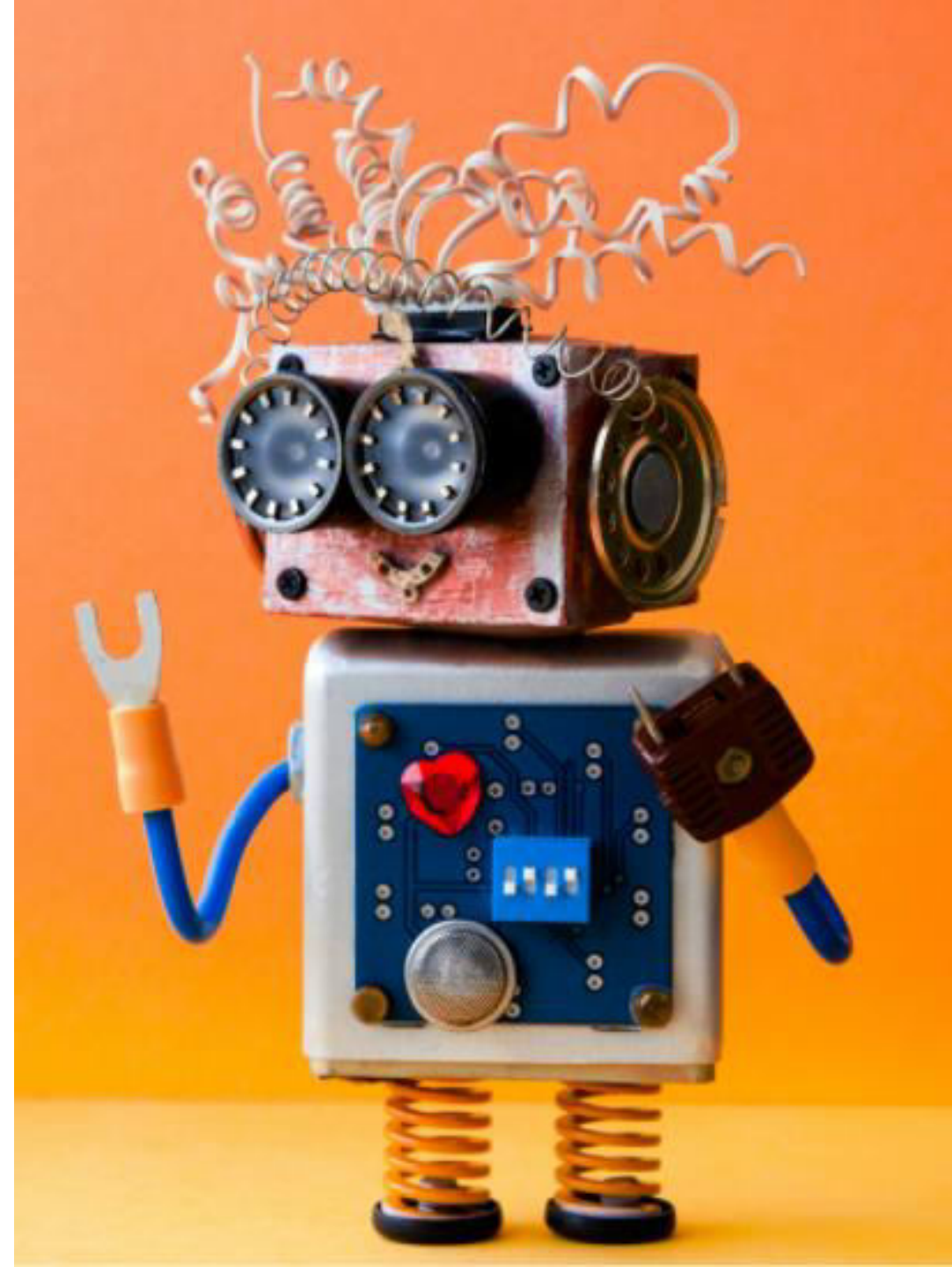
Name	Status	Type	Description
invoke-lambda-demo	Enabled	Scheduled Standard	

DEMO: LAMBDA INVOCATION WITH EVENTBRIDGE

GO BACK TO THE LAMBDA FUNCTION AND CLICK ON MONITOR



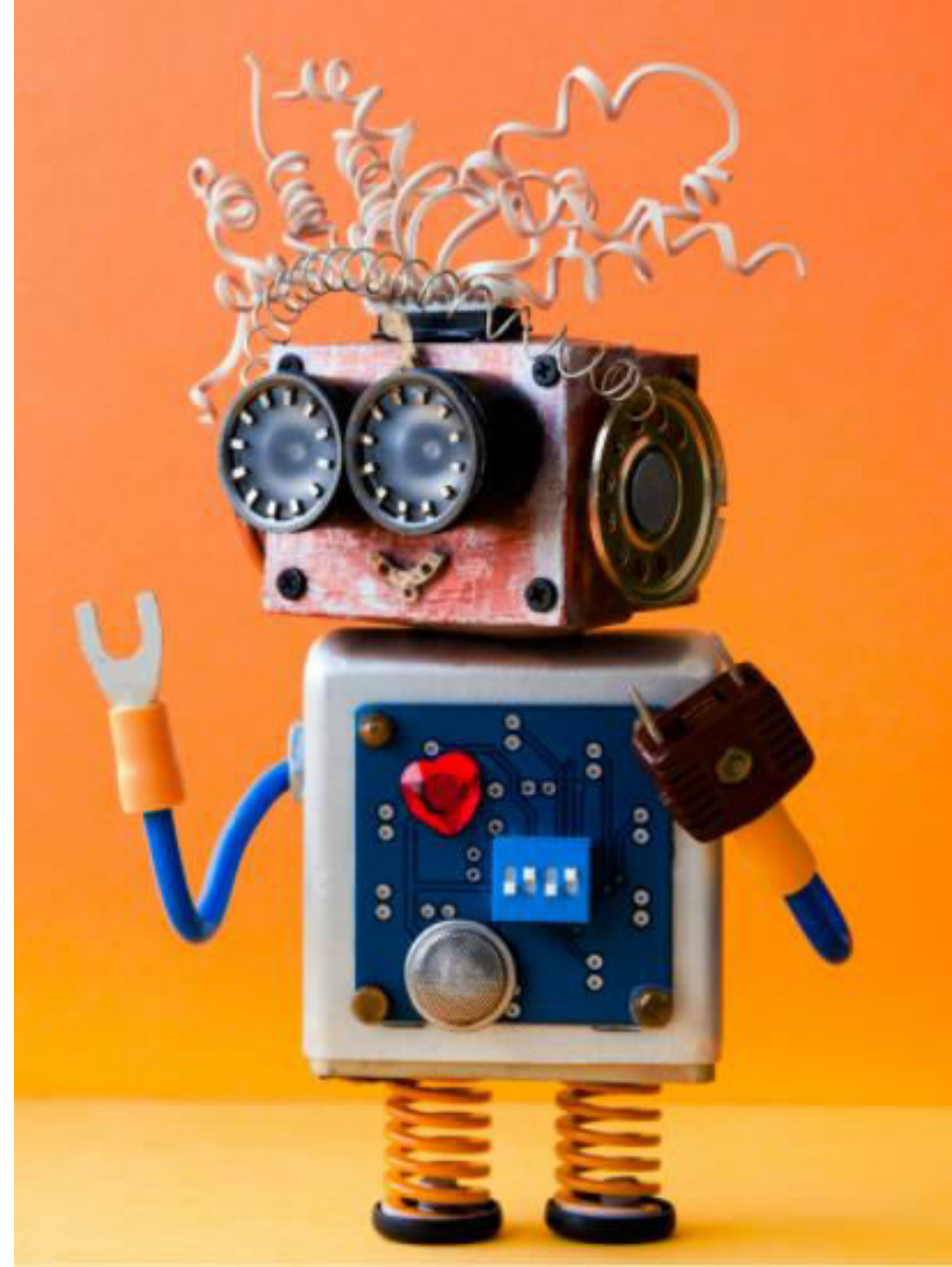
FINAL END-OF-DAY CAPSTONE PROJECT



FINAL CAPSTONE PROJECT

- Using boto3 SDK, define a Lambda Function named “FundsCalculator” that takes in the number of stock units and stock price and calculates the total value of the portfolio.
- Develop a script that consumes the total number of stocks and price of stocks from users.
- Return the total value of the portfolio.
- Configure the following test events using AWS Boto3 SDK:
 - Stock Units = 20, stock value = \$1000
 - Stock Units = 5, stock value = \$2000
- Monitor the logs in CloudWatch and ensure that the Lambda Function execution was successful.

FINAL END-OF-DAY CAPSTONE PROJECT SOLUTION



DEMO: LAMBDA INVOCATION WITH SDK

TASK #1. DEFINE A LAMBDA FUNCTION

```
In [1]: %%writefile lambda_function.py

import json
import uuid

def lambda_handler(event, context):

    # Read the input parameters
    count = event['StockUnits']
    price = event['StockPrice']

    # Calculate the total dollar value
    total = count * price

    # Return the result
    return {
        'TotalFunds': total
    }
```

Writing lambda_function.py

```
In [3]: # boto3 is the Amazon Web Services (AWS) Software Development Kit (SDK) for Python
# boto3 allows Python developer to write software that makes use of services like Amazon S3 and Amazon EC2
import boto3
import json
client = boto3.client('lambda')

# You must zip up the code of the Lambda function at some point
# so let's import ZipFile Module
# This module provides tools to create, read, write, append, and list a ZIP file.
from zipfile import ZipFile
with ZipFile('lambda.zip', 'w') as f:
    f.write('lambda_function.py')

# If submitting as a ZipFile, you need to insert raw data.
with open('lambda.zip', 'rb') as f:
    zipped_code = f.read()

# Note that SageMaker Notebooks don't have access to Lambda by default
# so you need to give permission to this SageMaker Notebook to access Lambda
response = client.create_function(
    FunctionName = 'FundsCalculator',
    Runtime = 'python3.8',
    Handler = 'lambda_function.lambda_handler',
    Code = dict(zipFile=zipped_code),
    Timeout = 60,
    MemorySize = 512,
    Role = 'arn:aws:iam::422132866096:role/service-role/My-First-Lambda-role-ozpjjvwf'
)
```

TASK #2. INVOKE A LAMBDA FUNCTION

```
In [5]: # Obtain How many stocks and the number of stocks from the bank customer

stock_price = int(input('What is the unit price of the stock you would like to buy'))
stock_count = int(input('How many stocks units you would like to purchase?'))

response = client.invoke(
    FunctionName = 'FundsCalculator',
    InvocationType = 'RequestResponse',
    LogType = 'Tail',
    Payload = json.dumps({'StockUnits' : stock_count, 'StockPrice' : stock_price}).encode('utf-8'),
)
```

What is the unit price of the stock you would like to buy?20
How many stocks units you would like to purchase?30

```
In [6]: print(response)
```

```
{'ResponseMetadata': {'RequestId': 'c87fab28-d3d8-4822-98fc-0b7532c47bc6', 'HTTPStatusCode': 200, 'Headers': {'date':  
    'Tue, 15 Feb 2022 22:20:05 GMT', 'content-type': 'application/json', 'content-length': '19', 'connection': 'keep-alive', 'x-amzn-requestid': 'c87fab28-d3d8-4822-98fc-0b7532c47bc6', 'x-amzn-remapped-content-length': '0', 'x-amz-executed-version':  
    '$LATEST', 'x-amz-log-result': 'U1RBUlQumVxdWZdeElkoibJoddmYwIyOCkM2Q4LTQ4MjItOTThmyyOwYjc1MzJJNDidvYzYKukVQTlUIUF1lcXVl3CRJDogvgzg3ZFmgjtZDnkoce00DYtL  
    K4zmHbtG3INTHyYq3QMZYCURicmfF0aw9uoiaYOS4MXCBtcwlCaixSzQGHRhyYXRpb2gIdGV1biBweSBTExploIA1MTTUJTfIWFIeIlbwgyeSBVC  
    2VKoiA0MCBNcgJlbmlER1cmF0aw9uoiaYnzguITybGMJCg==', 'x-amzn-trace-id': 'root=1-62ec2715-4dcac7ed3db9583a59b7c7;sampled=  
    0'}, 'RetryAttempts': 0}, 'Status': 200, 'LogResult': 'U1RBUlQumVxdWZdeElkoibJoddmYwIyOCkM2Q4LTQ4MjItOTThmyyOwYjc1MzJJNDidvYzYKukVQTlUIUF1lcXVl3CRJDogvgzg3ZFmgjtZDnkoce00DYtLTK4zmHTG3INTHyYq3QMZYCURicmfF0aw9uoiaYOS4MXCBtcwlCaixSzQGHRhyYXRpb2gIdGV1biBweSBTExploIA1MTTUJTfIWFIeIlbwgyeSBVC2VKoiA0MCBNcgJlbmlER1cmF0aw9uoiaYnzguITybGMJCg==', 'ExecutedVersion': '$LATEST', 'Payload': <  
    botocore.response.StreamingBody object at 8x7fa19084f438>
```

```
In [7]: data = response['Payload'].read()
```

```
In [8]: print(data)
```