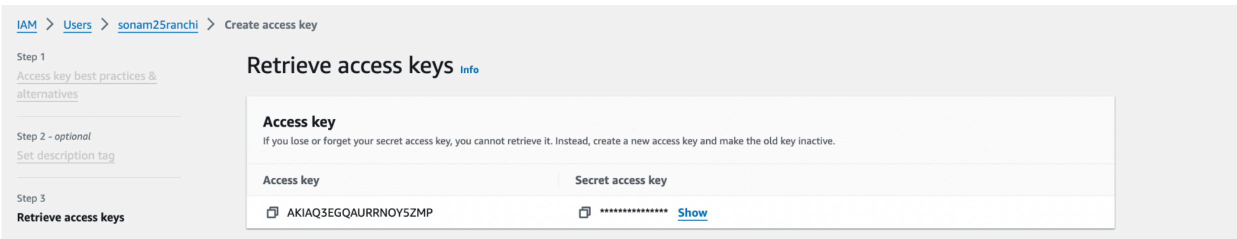
Name: Sonam Kumari
Employee ID: TAS204

AWS ASSIGNMENT

Executing the 'aws configure' command to set up and configure the AWS CLI (Command Line Interface) by providing the access key ID and secret access key.

We can find the access keys from AWS console >> IAM >> Users



Command: aws configure

```
sonamkumari@Sonams-MacBook-Air ~ % aws configure
AWS Access Key ID [****************LQTJ]: AKIAQ3EGQAUR7ATNVKH6
AWS Secret Access Key [****************DeZF]: 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
Default region name [us-east-1]: us-east-1
Default output format [json]: json
```

1. **Create S3 bucket from AWS CLI**
   a. To create the IAM Role with S3 FullAccess

   Step 1: Create the role
   Command: aws iam create-role --role-name ASSIGNMENTROLE --assume-role-policy-document file:///Users/sonamkumari/Desktop/trust-policy.json

   Trust-policy.json is a json file which specifies the trust relationship allowing EC2 to assume this role.
   trust-policy.json>>

```
{} trust-policy.json > ...
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "Service": "lambda.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
      }
    ]
  }
```

```
sonamkumari@Sonams-MacBook-Air ~ % aws iam create-role --role-name ASSIGNMENTROLE --assume-role-policy-document file:///Users/
sonamkumari/Desktop/trust-policy.json
{
    "Role": {
        "Path": "/",
        "RoleName": "ASSIGNMENTROLE",
        "RoleId": "AROAQ3EGQAURYBCIAIQCG",
        "Arn": "arn:aws:iam::058264126755:role/ASSIGNMENTROLE",
        "CreateDate": "2024-05-02T11:46:19+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "lambda.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }
    }
}
```

Step 2. To attach the S3 full access policy to the role
command: aws iam attach-role-policy --role-name ASSIGNMENTROLE --policy-arn arn:aws:iam::aws:policy/AmazonS3FullAccess

```
sonamkumari@Sonams-MacBook-Air ~ % aws iam attach-role-policy --role-name ASSIGNMENTROLE --policy-arn arn:aws:iam::aws:policy/
AmazonS3FullAccess
sonamkumari@Sonams-MacBook-Air ~ %
```

b. **Create an EC2 instance with above role**

Step 1: Creating Instance Profile
command: aws iam create-instance-profile --instance-profile-name EC2S3FullAccessProfile

Step 2: Adding instance profile to the role
command: aws iam add-role-to-instance-profile --instance-profile-name EC2S3FullAccessProfile --role-name ASSIGNMENTROLE

Step 3:To list all the instance profile
command: aws iam list-instance-profiles

```
sonamkumari@Sonams-MacBook-Air ~ % aws iam list-instance-profiles

{
    "InstanceProfiles": [
        {
            "Path": "/",
            "InstanceProfileName": "AmazonEMR-InstanceProfile-20240417T154415",
            "InstanceProfileId": "AIPAQ3EGQAUR2Z2GY7LPC",
            "Arn": "arn:aws:iam::058264126755:instance-profile/AmazonEMR-InstanceProfile-20240417T154415",
            "CreateDate": "2024-04-17T10:14:18+00:00",
            "Roles": [
                {
                    "Path": "/service-role/",
                    "RoleName": "AmazonEMR-InstanceProfile-20240417T154415",
                    "RoleId": "AROAQ3EGQAUR2R72JRBDU",
                    "Arn": "arn:aws:iam::058264126755:role/service-role/AmazonEMR-InstanceProfile-20240417T154415",
                    "CreateDate": "2024-04-17T10:14:17+00:00",
                    "AssumeRolePolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {
                                    "Service": "ec2.amazonaws.com"
                                },
                                "Action": "sts:AssumeRole"
                            }
                        ]
                    }
                }
            ]
        },
        {
            "Path": "/",
            "InstanceProfileName": "EC2S3FullAccessProfile",
            "InstanceProfileId": "AIPAQ3EGQAUR2Z6HFDFYL",
            "Arn": "arn:aws:iam::058264126755:instance-profile/EC2S3FullAccessProfile",
            "CreateDate": "2024-05-01T20:04:33+00:00",
            "Roles": [
                {
                    "Path": "/",
                    "RoleName": "ASSIGNMENTROLE",
                    "RoleId": "AROAQ3EGQAURYBCIAIQCG",
                    "Arn": "arn:aws:iam::058264126755:role/ASSIGNMENTROLE",
                    "CreateDate": "2024-05-02T11:46:19+00:00",
                    "AssumeRolePolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {
                                    "Service": "lambda.amazonaws.com"
                                },
                                "Action": "sts:AssumeRole"
                            }
                        ]
                    }
                }
            ]
        }
    ]
```

Step 4: Launch an EC2 instance using the AWS CLI and specify the IAM role
command: aws ec2 run-instances --image-id ami-04e5276ebb8451442 --instance-type t2.micro --iam-instance-profile Name=EC2S3FullAccessProfile

```
sonamkumari@Sonams-MacBook-Air ~ % aws ec2 run-instances --image-id ami-04e5276ebb8451442 --instance-type t2.micro
--iam-instance-profile Name=EC2S3FullAccessProfile

{
    "Groups": [],
    "Instances": [
        {
            "AmiLaunchIndex": 0,
            "ImageId": "ami-04e5276ebb8451442",
            "InstanceId": "i-01621f3a6a87ef901",
            "InstanceType": "t2.micro",
            "LaunchTime": "2024-05-02T12:09:59+00:00",
            "Monitoring": {
                "State": "disabled"
            },
            "Placement": {
                "AvailabilityZone": "us-east-1b",
                "GroupName": "",
                "Tenancy": "default"
            },
            "PrivateDnsName": "ip-172-31-19-255.ec2.internal",
            "PrivateIpAddress": "172.31.19.255",
            "ProductCodes": [],
            "PublicDnsName": "",
            "State": {
                "Code": 0,
                "Name": "pending"
            },
            "StateTransitionReason": "",
            "SubnetId": "subnet-0e5a39ec311195a05",
            "VpcId": "vpc-064bdb01ad4d5a20f",
            "Architecture": "x86_64",
            "BlockDeviceMappings": [],
            "ClientToken": "8bf8a19a-643c-46a6-9dad-bbc0e6244e4f",
            "EbsOptimized": false,
            "EnaSupport": true,
            "Hypervisor": "xen",
            "IamInstanceProfile": {
                "Arn": "arn:aws:iam::058264126755:instance-profile/EC2S3FullAccessProfile",
                "Id": "AIPAQ3EGQAUR2Z6HFDFYL"
            },
            "NetworkInterfaces": [
                {
                    "Attachment": {
```

c. **Create a Bucket from AWS CLI**

command: aws s3 mb s3://sonam25ranchibucket

```
sonamkumari@Sonams-MacBook-Air ~ % aws s3 mb s3://sonam25ranchibucket

make_bucket: sonam25ranchibucket
sonamkumari@Sonams-MacBook-Air ~ %
```

2. **Put files in S3 bucket from lambda**
    a. **Create custom role for AWS lambda which will have only put object access**
    Step 1: Create a Lambda role
    command: aws iam create-role --role-name LambdaS3PutRole
    --assume-role-policy-document
    file:///Users/sonamkumari/Desktop/lambda-trust-policy.json

```
sonamkumari@Sonams-MacBook-Air ~ % aws iam create-role --role-name LambdaS3PutRole --assume-role-policy-document
file:///Users/sonamkumari/Desktop/lambda-trust-policy.json
{
    "Role": {
        "Path": "/",
        "RoleName": "LambdaS3PutRole",
        "RoleId": "AROAQ3EGQAUR7Z6OOR6ZE",
        "Arn": "arn:aws:iam::058264126755:role/LambdaS3PutRole",
        "CreateDate": "2024-05-02T18:36:47+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "lambda.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }
    }
}
sonamkumari@Sonams-MacBook-Air ~ %
```

Step 2: Attach limited permissions to put object in S3
command: aws iam put-role-policy --role-name LambdaS3PutRole --policy-name PutObjectPolicy --policy-document file://s3-put-policy.json

```
Users > sonamkumari > Desktop > {} s3-put-policy.json > ...
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "s3:PutObject"
        ],
        "Resource": [
          "arn:aws:s3:::sonam25ranchibucket/*"
        ]
      }
    ]
  }
```

```
sonamkumari@Sonams-MacBook-Air ~ % aws iam put-role-policy --role-name LambdaS3PutRole --policy-name PutObjectPol
icy --policy-document file:///Users/sonamkumari/Desktop/s3-put-policy.json
sonamkumari@Sonams-MacBook-Air ~ %
```

b. **Add Role to Generate and Access CloudWatch Logs**

Step 1: Creating role

```
sonamkumari@Sonams-MacBook-Air ~ % aws iam attach-role-policy --role-name LambdaS3PutRole --policy-arn arn:aws:ia
m::aws:policy/service-role/AWSLambdaBasicExecutionRole

sonamkumari@Sonams-MacBook-Air ~ %
```

Step 2: Creating policy lambda-cloudwatch-policy.json

Code for lambda-cloudwatch-policy.json :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    }
  ]
}
```

```
sonamkumari@Sonams-MacBook-Air ~ % aws iam create-policy --policy-name LambdaCloudWatchLogsPolicyNew --policy-document file:///Users/
amkumari/Desktop/lambda-cloudwatch-policy.json

{
    "Policy": {
        "PolicyName": "LambdaCloudWatchLogsPolicyNew",
        "PolicyId": "ANPAQ3EGQAUR33SV32H22",
        "Arn": "arn:aws:iam::058264126755:policy/LambdaCloudWatchLogsPolicyNew",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2024-05-03T07:32:45+00:00",
        "UpdateDate": "2024-05-03T07:32:45+00:00"
    }
}
```

Step 3: creating s3PutObjectPolicy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutRule",
        "events:PutTargets"
      ],
      "Resource": "*"
    }
  ]
}
```

```
sonamkumari@Sonams-MacBook-Air ~ % aws iam create-policy --policy-name s3PutObjectPolicy --policy-document file:///Users/sonamkumari/Des
ktop/s3-put-policy.json

{
    "Policy": {
        "PolicyName": "s3PutObjectPolicy",
        "PolicyId": "ANPAQ3EGQAURT3DVEUPEF",
        "Arn": "arn:aws:iam::058264126755:policy/s3PutObjectPolicy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2024-05-03T07:34:43+00:00",
        "UpdateDate": "2024-05-03T07:34:43+00:00"
    }
}
sonamkumari@Sonams-MacBook-Air ~ %
```

Step 4: Attaching the above two policy with the role created in step 1

```
sonamkumari@Sonams-MacBook-Air ~ % aws iam attach-role-policy --role-name newlambdaS3PutRole --policy-arn arn:aws:iam::058264126755:poli
cy/s3PutObjectPolicy
sonamkumari@Sonams-MacBook-Air ~ % aws iam attach-role-policy --role-name newlambdaS3PutRole --policy-arn arn:aws:iam::058264126755:poli
cy/LambdaCloudWatchLogsPolicyNew
sonamkumari@Sonams-MacBook-Air ~ %
```

c. **In python script, generate json in given format and save .json file in bucket created**

Python code snippet for AWS Lambda:

Users > sonamkumari > Desktop > import_json.py > ...

```python
import json
import boto3
import os
from datetime import datetime

def lambda_handler(event, context):
    # Get current execution count from environment variable
    execution_count = int(os.environ.get('EXECUTION_COUNT', 0))

    # Increment execution count
    execution_count += 1

    # Update execution count in environment variable
    os.environ['EXECUTION_COUNT'] = str(execution_count)

    # Stop execution after 3 runs
    if execution_count > 3:
        return {
            'statusCode': 200,
            'body': 'Execution stopped after 3 runs'
        }

    # Generate JSON data
    json_data = {
        "transaction_id": 12345,
        "payment_mode": "card/netbanking/upi",
        "amount": 200.0,
        "customer_id": 101,
        "timestamp": str(datetime.now())
    }

    # Save JSON data to S3
    bucket_name = 'sonam25ranchibucket'
    file_name = f"transaction_{datetime.now().strftime('%Y-%m-%d_%H-%M-%S')}.json"

    s3 = boto3.client('s3')
    s3.put_object(Bucket=bucket_name, Key=file_name, Body=json.dumps(json_data))

    return {
        'statusCode': 200,
        'body': json.dumps('JSON file saved successfully')
    }
```

Create a zip file for the above file and executing CLI command to create function with name 'my-lambda-function'

```
sonamkumari@Sonams-MacBook-Air ~ % aws lambda create-function \
              --function-name my-lambda-function \
     --runtime python3.8 \
              --role arn:aws:iam::058264126755:role/newlambdaS3PutRole \
              --handler lambda_function.lambda_handler \
              --zip-file fileb:///Users/sonamkumari/Desktop/import_json.py.zip
{
    "FunctionName": "my-lambda-function",
    "FunctionArn": "arn:aws:lambda:us-east-1:058264126755:function:my-lambda-function",
    "Runtime": "python3.8",
    "Role": "arn:aws:iam::058264126755:role/newlambdaS3PutRole",
    "Handler": "lambda_function.lambda_handler",
    "CodeSize": 1047,
    "Description": "",
    "Timeout": 3,
    "MemorySize": 128,
    "LastModified": "2024-05-03T07:52:49.463+0000",
    "CodeSha256": "/Bfo/fr/ElbZ20Dfc38gYGiYsEeRDOgkApxZPkU3ug8=",
    "Version": "$LATEST",
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "RevisionId": "ac277a2c-263e-4a90-a2f8-860caa59e928",
    "State": "Pending",
    "StateReason": "The function is being created.",
    "StateReasonCode": "Creating",
    "PackageType": "Zip",
    "Architectures": [
        "x86_64"
    ],
    "EphemeralStorage": {
        "Size": 512
    },
    "SnapStart": {
        "ApplyOn": "None",
        "OptimizationStatus": "Off"
    },
    "RuntimeVersionConfig": {
        "RuntimeVersionArn": "arn:aws:lambda:us-east-1::runtime:830ab1165f1cdc47b8ae1e516f3936d8a176aec50d0699efd25a8bfeabb797b4"
    },
    "LoggingConfig": {
        "LogFormat": "Text",
        "LogGroup": "/aws/lambda/my-lambda-function"
    }
}
sonamkumari@Sonams-MacBook-Air ~ %
```

d. Schedule the Job to Run Every Minute and Stop After 3 Runs

Step 1: Create a CloudWatch Event Rule to trigger the Lambda every minute
Command: aws events put-rule --name "LambdaExecutionScheduleRule" --schedule-expression "rate(1 minute)" --state "ENABLED" --description "Trigger Lambda function every minute"

```
sonamkumari@Sonams-MacBook-Air ~ % aws events put-rule --name "LambdaExecutionScheduleRule" --schedule-expression "rate(1 minute)" --sta
te "ENABLED" --description "Trigger Lambda function every minute"
{
    "RuleArn": "arn:aws:events:us-east-1:058264126755:rule/LambdaExecutionScheduleRule"
}
sonamkumari@Sonams-MacBook-Air ~ %
```

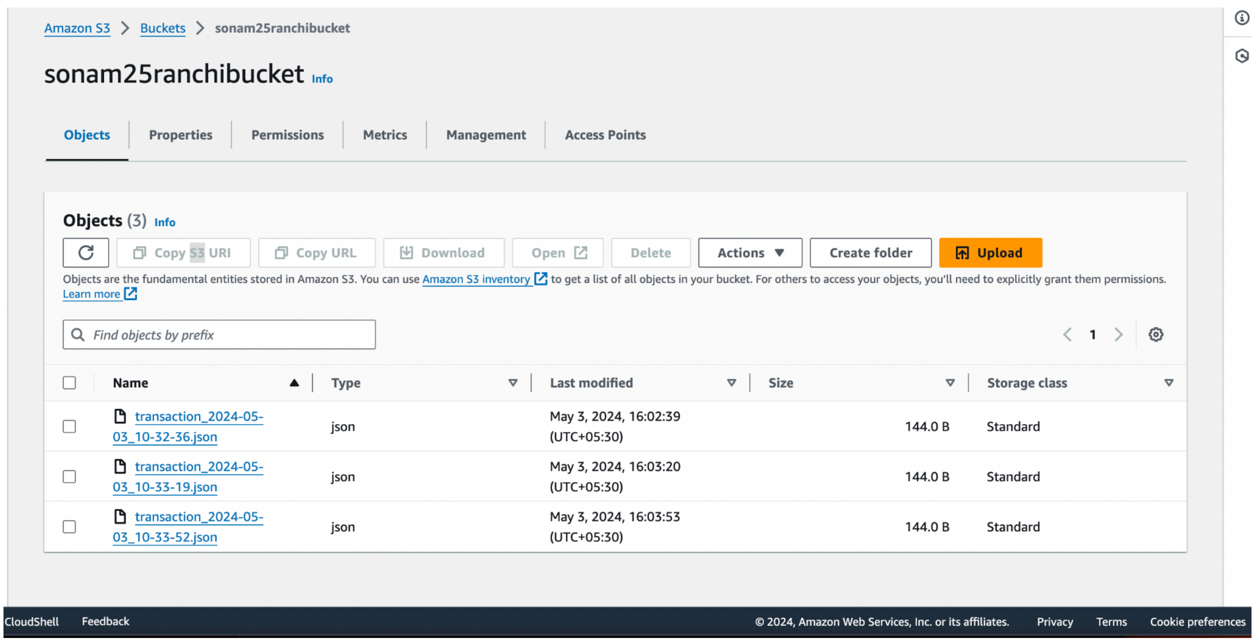
Step 2 : Add permissions to the Lambda function to allow it to be invoked by the CloudWatch Event
Command: aws events put-targets --rule "LambdaExecutionScheduleRule" --targets "Id"="1","Arn"="arn:aws:lambda:us-east-1:058264126755:function:my-lambda-function"

```
sonamkumari@Sonams-MacBook-Air ~ % aws events put-targets --rule "LambdaExecutionScheduleRule" --targets "Id"="1","Arn"="arn:aws:lambda:
us-east-1:058264126755:function:my-lambda-function"
{
    "FailedEntryCount": 0,
    "FailedEntries": []
}
sonamkumari@Sonams-MacBook-Air ~ %
```

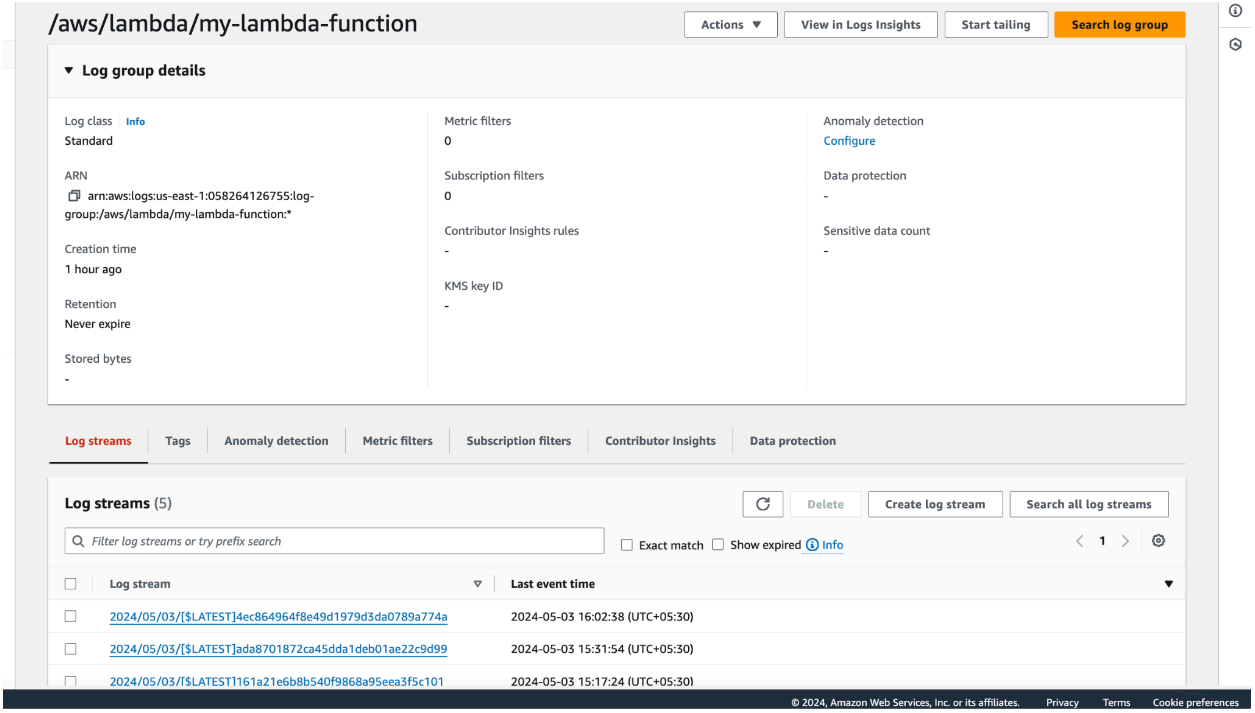
Step 3:Invoking the function through CLI
Command: aws lambda invoke --function-name my-lambda-function --payload '{}' output.json

```
sonamkumari@Sonams-MacBook-Air ~ % aws lambda invoke --function-name my-lambda-function --payload '{}' output.json
{
    "StatusCode": 200,
    "ExecutedVersion": "$LATEST"
}
```

e. Check if cloudwatch logs are generated



logs>>



3. **API gateway- Lambda Integration**

   a. **Modify lambda function to accept parameters and return file name.**

      Step 1: Modifying the lambda function
      Code:

```
Import json
import boto3
from datetime import datetime

def lambda_handler(event, context):
    # Get parameters from the request body
    transaction_id = event.get('transaction_id')
    payment_mode = event.get('payment_mode')
    amount = event.get('amount')
    customer_id = event.get('customer_id')

    # Check if required parameters are present
    if not all([transaction_id, payment_mode, amount, customer_id]):
        return {
            'statusCode': 400,
            'body': json.dumps({'error': 'Missing required parameters'})
```

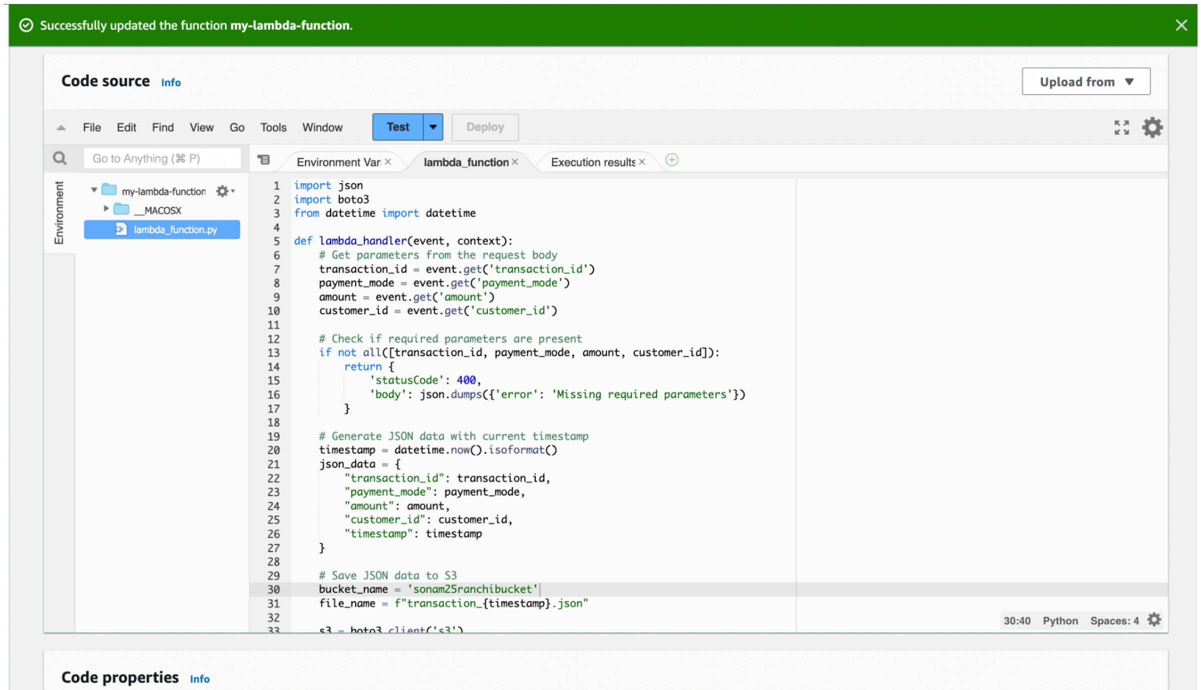
```
    }

    # Generate JSON data with current timestamp
    timestamp = datetime.now().isoformat()
    json_data = {
        "transaction_id": transaction_id,
        "payment_mode": payment_mode,
        "amount": amount,
        "customer_id": customer_id,
        "timestamp": timestamp
    }

    # Save JSON data to S3
    bucket_name = 'sonam25ranchibucket'
    file_name = f"transaction_{timestamp}.json"

    s3 = boto3.client('s3')
    s3.put_object(Bucket=bucket_name, Key=file_name,
Body=json.dumps(json_data))

    return {
        'statusCode': 200,
        'body': json.dumps({'file_name': file_name})
    }
```



Step 2: Invoking the function to return file name

```
sonamkumari@Sonams-MacBook-Air ~ % aws lambda invoke --function-name my-lambda-function --cli-binary-format raw-in-base64-out  --payload
 '{"transaction_id": 12345, "payment_mode": "card/netbanking/upi", "amount": 200.0,"customer_id": 101}' output.json
{
    "StatusCode": 200,
    "ExecutedVersion": "$LATEST"
}
sonamkumari@Sonams-MacBook-Air ~ % cat output.json
{"statusCode": 200, "body": "{\"file_name\": \"transaction_2024-05-03T10:53:54.932058.json\"}"}%
sonamkumari@Sonams-MacBook-Air ~ %
```

b. Create a POST API from API Gateway, pass parameters as request body to Lambda job. Return filename and status code as response.

Step 1: Create the REST API
Command: aws apigateway create-rest-api --name 'MyFileAPI' --region us-east-1

Step 2: Get the root resource id
Command: aws apigateway get-resources --rest-api-id c8b3xd17a5 --region us-east-1

Note: Api-id is generated in the step 1

Step 3: Create a Resource
Command: aws apigateway create-resource --rest-api-id c8b3xd17a5 --parent-id xkyxyecjs8 --path-part files --region us-east-1

```
sonamkumari@Sonams-MacBook-Air ~ % aws apigateway create-rest-api --name 'MyFileAPI' --region us-east-1

{
    "id": "c8b3xd17a5",
    "name": "MyFileAPI",
    "createdDate": "2024-05-03T16:33:58+05:30",
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
        "types": [
            "EDGE"
        ]
    },
    "disableExecuteApiEndpoint": false,
    "rootResourceId": "xkyxyecjs8"
}
sonamkumari@Sonams-MacBook-Air ~ % aws apigateway get-resources --rest-api-id c8b3xd17a5 --region us-east-1

{
    "items": [
        {
            "id": "xkyxyecjs8",
            "path": "/"
        }
    ]
}
sonamkumari@Sonams-MacBook-Air ~ % aws apigateway create-resource --rest-api-id c8b3xd17a5 --parent-id xkyxyecjs8 --path-part files --r
gion us-east-1

{
    "id": "9foytm",
    "parentId": "xkyxyecjs8",
    "pathPart": "files",
    "path": "/files"
}
```

Step 4: Create a POST method

Command: aws apigateway put-method --rest-api-id c8b3xd17a5 --resource-id xkyxyecjs8 --http-method POST --authorization-type NONE --region us-east-1

```
sonamkumari@Sonams-MacBook-Air ~ % aws apigateway put-method --rest-api-id c8b3xd17a5 --resource-id xkyxyecjs8 --http-method POST --auth
orization-type NONE --region us-east-1

{
    "httpMethod": "POST",
    "authorizationType": "NONE",
    "apiKeyRequired": false
}
```

Step 5: Set the lambda integration

Command: aws apigateway put-integration --rest-api-id c8b3xd17a5 --resource-id xkyxyecjs8 --http-method POST --type AWS_PROXY --integration-http-method POST --uri 'arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-east-1:058264126755:function:MyLambdaFunction/invocations' --region us-east-1

```
sonamkumari@Sonams-MacBook-Air ~ % aws apigateway put-integration --rest-api-id c8b3xd17a5 --resource-id xkyxyecjs8 --http-method POS
-type AWS_PROXY --integration-http-method POST --uri 'arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-
st-1:058264126755:function:MyLambdaFunction/invocations' --region us-east-1

{
    "type": "AWS_PROXY",
    "httpMethod": "POST",
    "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-east-1:058264126755:function:MyLambdaFunc
n/invocations",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "timeoutInMillis": 29000,
    "cacheNamespace": "xkyxyecjs8",
    "cacheKeyParameters": []
}
```

Step 6: Deploy the API

Command: aws apigateway create-deployment --rest-api-id c8b3xd17a5 --stage-name prod --region us-east-1

```
sonamkumari@Sonams-MacBook-Air ~ % aws apigateway create-deployment --rest-api-id c8b3xd17a5 --stage-name prod --region us-east-1

{
    "id": "m0675l",
    "createdDate": "2024-05-03T16:54:59+05:30"
}
```

c. **Consume API from local machine and pass unique data to lambda.**

Command: curl -X POST 'https://c8b3xd17a5.execute-api.us-east-1.amazonaws.com/prod/my-lambda-function' \
-d '{"payload": {"Item": {"transaction_id": 12345, "payment_mode": "card/netbanking/upi", "amount": 200.0, "customer_id": 101}}}'
{"file_name": "transaction_2024-05-03T12:50:46.773398.json"}

```
sonamkumari@Sonams-MacBook-Air ~ % curl -X POST 'https://c8b3xd17a5.execute-api.us-east-1.amazonaws.com/prod/my-lambda-function' \
-d '{"payload": {"Item": {"transaction_id": 12345, "payment_mode": "card/netbanking/upi", "amount": 200.0, "customer_id": 101}}}'
{"statusCode": 200, "file_name": "transaction_2024-05-03T12:57:36.015164.json"}%
sonamkumari@Sonams-MacBook-Air ~ %
```

d. **Check if cloudwatch logs are generated**