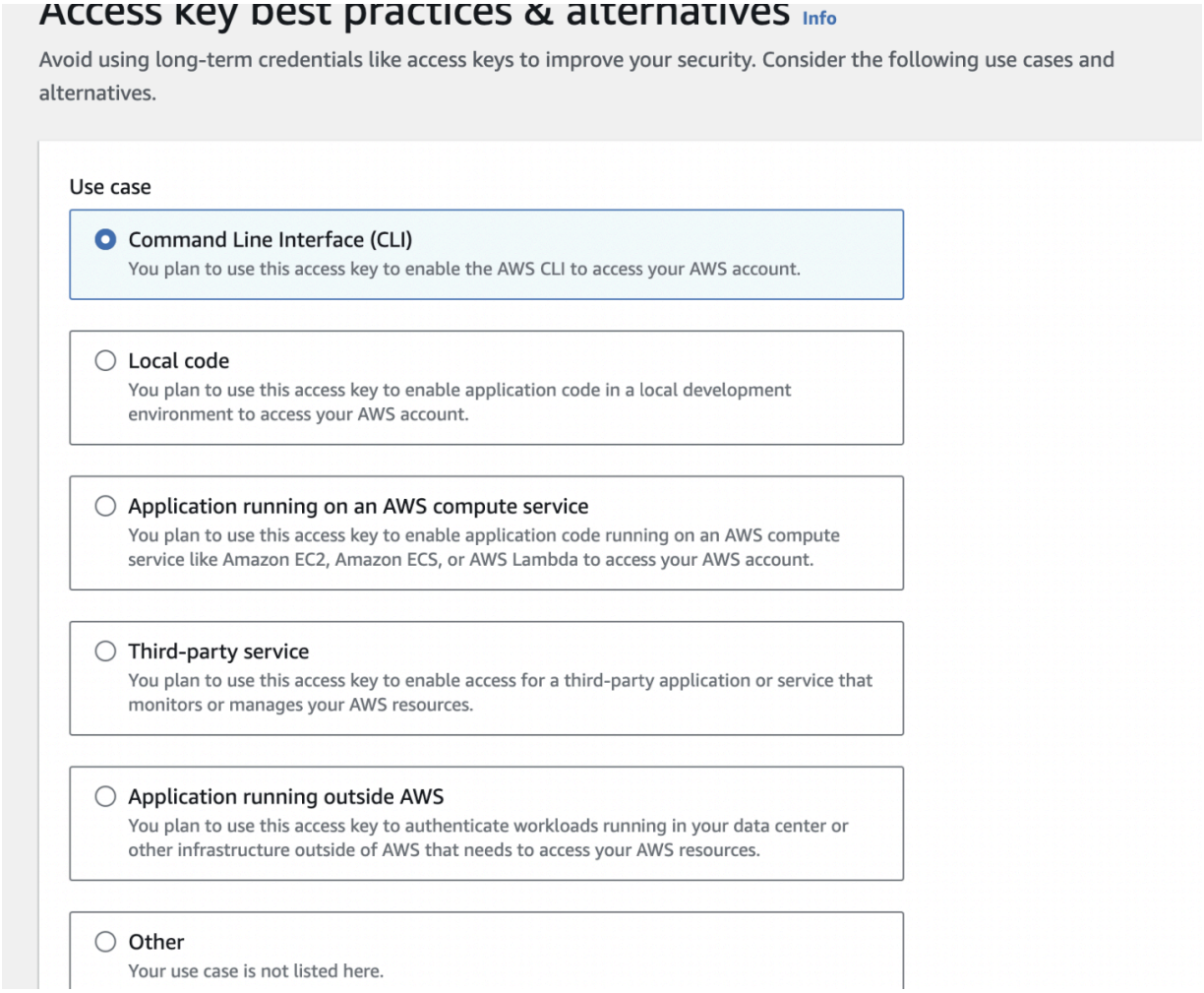
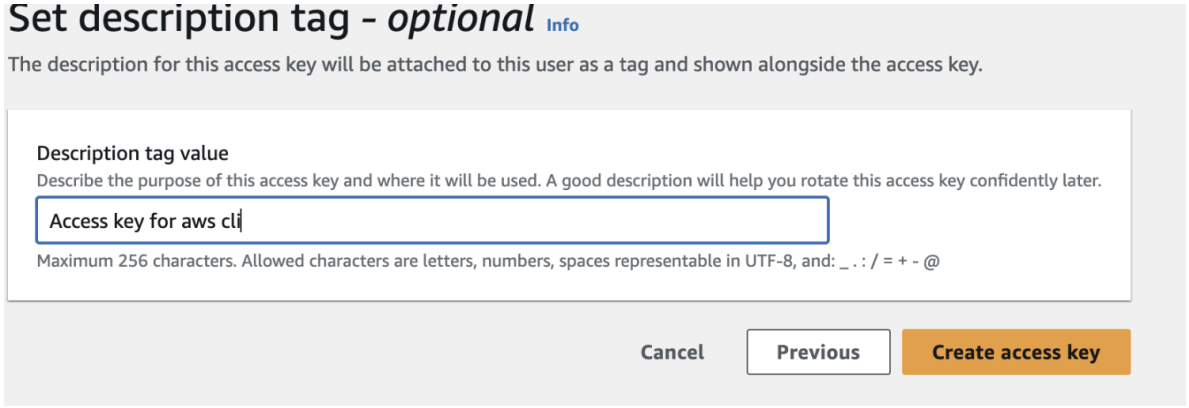
AWS Assignment

- Login to AWS CLI using an access key.





- Enter the user Access ID and Secret Access key

```
AWS Access Key ID [None]: AKIAQ3EGRH55WRAKSXOV
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json
```

- Create a role with trust policy to use EC2 instance
- Role name - devProjectRole

1. A. Create a S3 bucket in aws cli
    - Create a IAM role with s3 full access policy
    - Policy created
    - Attach the policy to the role

```
  --policy-name S3FullAccessPolicy \
  --policy-document file://s3FullAccessPolicy.json

{
    "Policy": {
        "PolicyName": "S3FullAccessPolicy",
        "PolicyId": "ANPAQ3EGRH557MQ4NK3QS",
        "Arn": "arn:aws:iam::058264207227:policy/S3FullAccessPolicy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2024-05-03T06:54:22Z",
        "UpdateDate": "2024-05-03T06:54:22Z"
    }
}
```

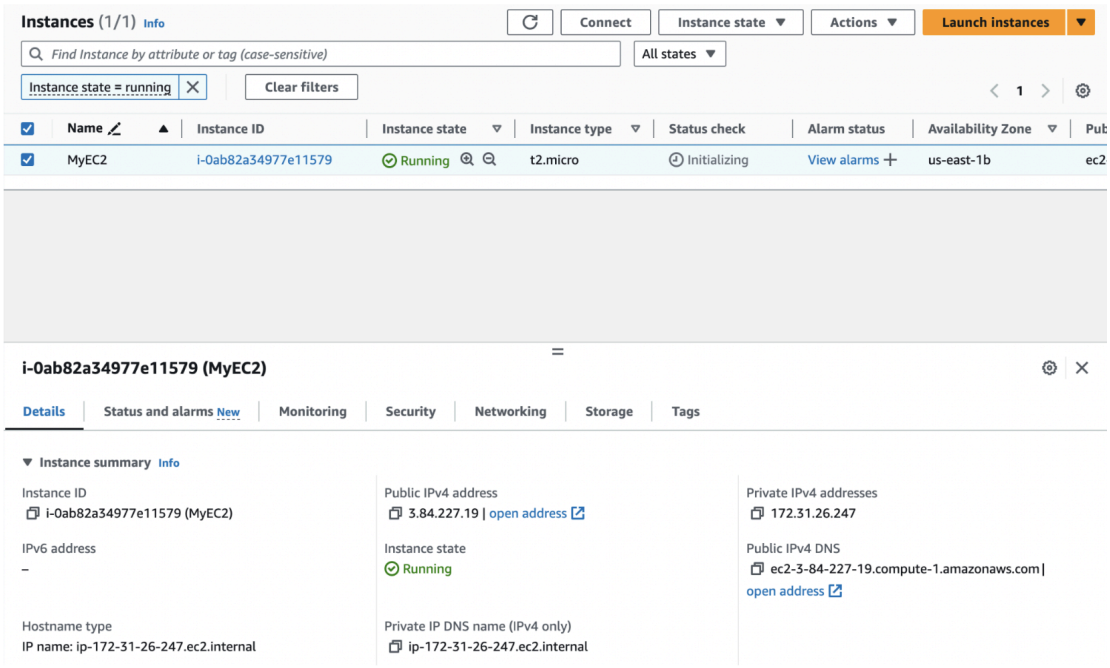
1. B. Create a EC2 instance with the above role
   - Create Instance profile
   - Create EC2 instance that will be used throughout the project
   - Pass the appropriate Amazon Machine Image, key Name and other required credentials.

```
{
    "InstanceProfile": {
        "Path": "/",
        "InstanceProfileName": "MyInstanceProfile",
        "InstanceProfileId": "AIPAQ3EGRH55YXAMGPBCX",
        "Arn": "arn:aws:iam::058264207227:instance-profile/MyInstanceProfile",
        "CreateDate": "2024-05-03T07:20:51Z",
        "Roles": []
    }
}
```

```
    --image-id ami-07caf09b362be10b8 \
    --instance-type t2.micro \
    --key-name projectKey \
    --iam-instance-profile Arn=arn:aws:iam::058264207227:instance-profile/MyInstanceProfile;

{
    "Groups": [],
    "Instances": [
        {
            "AmiLaunchIndex": 0,
            "ImageId": "ami-07caf09b362be10b8",
            "InstanceId": "i-0ab82a34977e11579",
            "InstanceType": "t2.micro",
            "KeyName": "projectKey",
            "LaunchTime": "2024-05-03T07:21:11.000Z",
            "Monitoring": {
                "State": "disabled"
            },
            "Placement": {
                "AvailabilityZone": "us-east-1b",
                "GroupName": "",
```

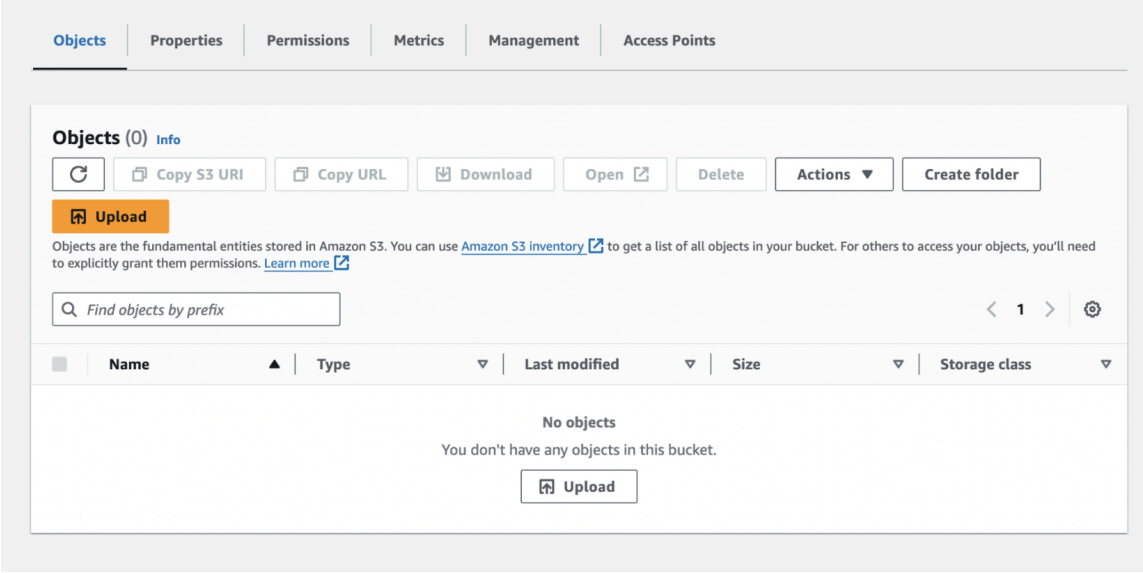
```
                "VirtualizationType": "hvm",
                "CpuOptions": {
                    "CoreCount": 1,
                    "ThreadsPerCore": 1
                },
                "CapacityReservationSpecification": {
                    "CapacityReservationPreference": "open"
                },
                "MetadataOptions": {
                    "State": "pending",
                    "HttpTokens": "required",
                    "HttpPutResponseHopLimit": 2,
                    "HttpEndpoint": "enabled",
                    "HttpProtocolIpv6": "disabled",
                    "InstanceMetadataTags": "disabled"
                },
                "EnclaveOptions": {
                    "Enabled": false
                },
                "BootMode": "uefi-preferred",
                "PrivateDnsNameOptions": {
                    "HostnameType": "ip-name",
                    "EnableResourceNameDnsARecord": false,
                    "EnableResourceNameDnsAAAARecord": false
                },
                "MaintenanceOptions": {
                    "AutoRecovery": "default"
                },
                "CurrentInstanceBootMode": "legacy-bios"
            }
        ],
        "OwnerId": "058264207227",
        "ReservationId": "r-046bea443b5677130"
```

- EC2 instance (MyEC2) successfully created and running.



1. C. Create a bucket from aws cli
   - Create bucket using CLI command
   - BucketName - devprojectbucket

Bucket successfully created.



2 . Put files in aws s3 using lambda.

2. A. Create a custom role for aws lambda which will have only put access

```
    --policy-name LambdaS3PutPolicy \
    --policy-document file://s3LamdaPolicy.json;

{
    "Policy": {
        "PolicyName": "LambdaS3PutPolicy",
        "PolicyId": "ANPAQ3EGRH55WG5PT2FXD",
        "Arn": "arn:aws:iam::058264207227:policy/LambdaS3PutPolicy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2024-05-03T07:42:41Z",
        "UpdateDate": "2024-05-03T07:42:41Z"
    }
}
```

```
    --role-name LambdaS3PutRole \
    --assume-role-policy-document '{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "lambda.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'
```

2. B. Add role to generate and access cloud watch logs.

```
    --policy-name LambdaS3CloudWatchPolicy \
    --policy-document file://cloudWatchPolicy.json

{
    "Policy": {
        "PolicyName": "LambdaS3CloudWatchPolicy",
        "PolicyId": "ANPAQ3EGRH55TWHIU46XI",
        "Arn": "arn:aws:iam::058264207227:policy/LambdaS3CloudWatchPolicy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2024-05-03T08:06:28Z",
        "UpdateDate": "2024-05-03T08:06:28Z"
    }
}
```

● Create the policy and attach the policy to the Lambda S3 Put Role.

2. C. In Python Script, generate json in given format and save.json file in bucket created.

● Create a AWS Lambda function using a python file.


```
{
    "FunctionName": "my-lambda-function",
    "FunctionArn": "arn:aws:lambda:us-east-1:058264207227:function:my-lambda-function",
    "Runtime": "python3.8",
    "Role": "arn:aws:iam::058264207227:role/LambdaS3PutRole",
    "Handler": "lambda_function.lambda_handler",
    "CodeSize": 735,
    "Description": "",
    "Timeout": 3,
    "MemorySize": 128,
    "LastModified": "2024-05-03T09:36:03.755+0000",
    "CodeSha256": "i758F6RdPdVbJ21A60dRXtMkTI7jSQMJ9i/qBuJX9tg=",
    "Version": "$LATEST",
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "RevisionId": "64b7bc2c-8ccb-4a45-9240-ba1d40330322",
    "State": "Pending",
    "StateReason": "The function is being created.",
    "StateReasonCode": "Creating",
    "PackageType": "Zip",
    "Architectures": [
        "x86_64"
    ],
    "EphemeralStorage": {
        "Size": 512
    },
    "SnapStart": {
        "ApplyOn": "None",
```


- Invoke a Lambda expression

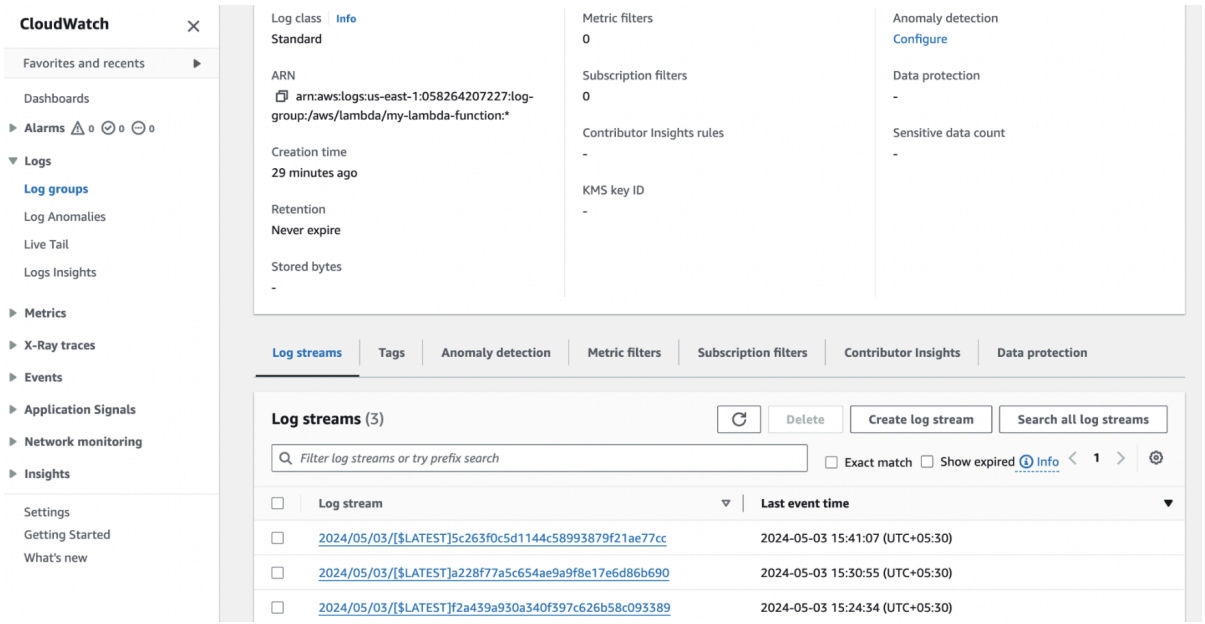

```
{
    "StatusCode": 200,
    "ExecutedVersion": "$LATEST"
}
```


- The lambda function code.

- Added str(tdatetime.now()) in the timestamp field so that each log will have its own custom timestamp.

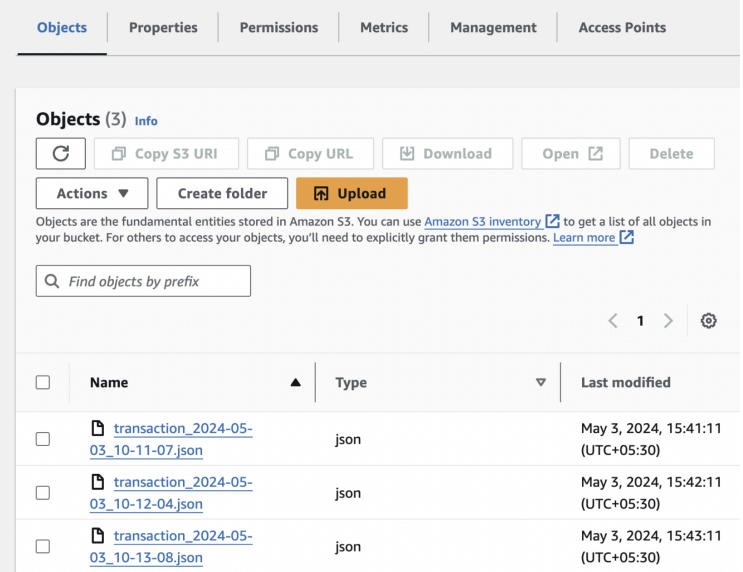
```python
import boto3
import os
from datetime import datetime

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
```

2. D. Schedule the job to run every three minutes and stop execution after 3 times.



2. E. Check if the cloud watch logs are generated.

- Logs are successfully pushed to the S3 bucket.



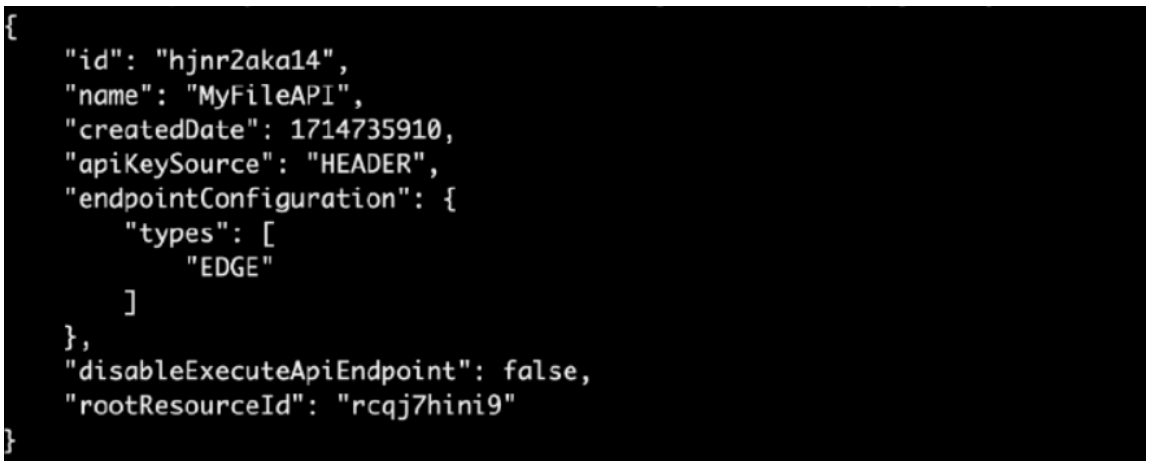
3. A. Modify lambda function to accept parameters and return fileName.

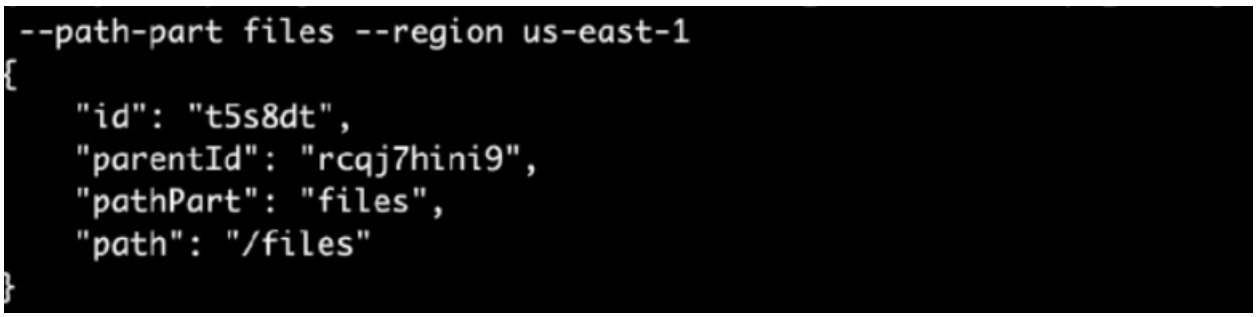
- Lambda code changed and function tested.

```
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
```

```python
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})
}
```

3. B. Create a POST api from gateway, pass parameters as request body to Lambda function. Return file name and status code as response.





- Created the API, connected the API with lambda.

3. C. Consume the API from local machine and pass unique data to lambda.

```
lambda-function' \
-d '{"operation": "create", "payload": {"Item":{ "transaction_id": 12345, "payment_mode": "card/netbanking/upi", "am
"customer_id": 101, "timestamp":"77" }}'
{"file_name": "transaction_2024-05-03T12:13:39.852779.json"}
```

```python
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')
    # return {'statusCode':200}

    if transaction_id!=None and payment_mode!=None and amount!=None and customer_id!=None:
        return {
            "statusCode":400,
            'body':'Missed'
        }

    # 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
```

- Lambda code appropriately changed.

3. D. Check if cloud watch logs are generated.