

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

The screenshot shows the 'Access key' section of the AWS IAM 'Retrieve access keys' page. It displays two sets of keys:

Access key	Secret access key
AKIAQ3EGQAURNOY5ZMP	***** Show

Command: aws configure

```
sonamkumari@Sonams-MacBook-Air ~ % aws configure
AWS Access Key ID [*****]: AKIAQ3EGQAUR7ATNVKH6
AWS Secret Access Key [*****]: 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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 > ...
1  {
2      "Version": "2012-10-17",
3      "Statement": [
4          {
5              "Effect": "Allow",
6              "Principal": {
7                  "Service": "lambda.amazonaws.com"
8              },
9              "Action": "sts:AssumeRole"
10         }
11     ]
12 }
13 }
```

```
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": "AROAQ3EGQAUR2R72JRBUD",
                    "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": "AIPAQ3EGQAUR2ZGHDYFL",
            "Arn": "arn:aws:iam::058264126755:instance-profile/EC2S3FullAccessProfile",
            "CreateDate": "2024-05-01T20:04:33+00:00",
            "Roles": [
                {
                    "Path": "/",
                    "RoleName": "ASSIGNMENTROLE",
                    "RoleId": "AROAQ3EGQARYBCIAIQCG",
                    "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": "AIPAQ3EGQAUR2Z6HFDYLY"
      },
      "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": "AROAQ3EGQAUR7Z600R6ZE",
    "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

```
1  {
2    "Version": "2012-10-17",
3    "Statement": [
4      {
5        "Effect": "Allow",
6        "Action": [
7          "s3:PutObject"
8        ],
9        "Resource": [
10          "arn:aws:s3:::sonam25ranchibucket/*"
11        ]
12      }
13    ]
14  }
```

```
sonamkumari@Sonams-MacBook-Air ~ % aws iam put-role-policy --role-name LambdaS3PutRole --policy-name PutObjectPolicy --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:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
sonamkumari@Sonams-MacBook-Air ~ %
```

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

Code for lambda-cloudwatch-policy.json :

```
1  {
2      "Version": "2012-10-17",
3      "Statement": [
4          {
5              "Effect": "Allow",
6              "Action": [
7                  "logs:CreateLogGroup",
8                  "logs:CreateLogStream",
9                  "logs:PutLogEvents"
10             ],
11            "Resource": "*"
12        }
13    ]
14 }
15 }
```

```
sonamkumari@Sonams-MacBook-Air ~ % aws iam create-policy --policy-name LambdaCloudWatchLogsPolicyNew --policy-document file:///Users/sonamkumari/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

```
1  {
2      "Version": "2012-10-17",
3      "Statement": [
4          {
5              "Effect": "Allow",
6              "Action": [
7                  "events:PutRule",
8                  "events:PutTargets"
9              ],
10             "Resource": "*"
11        }
12    ]
13 }
```

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

```
{
  "Policy": {
    "PolicyName": "s3PutObjectPolicy",
    "PolicyId": "ANPAQ3EGQAUR3DVEUPEF",
    "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:policy/s3PutObjectPolicy
sonamkumari@Sonams-MacBook-Air ~ % aws iam attach-role-policy --role-name newlambdaS3PutRole --policy-arn arn:aws:iam::058264126755:policy/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 > ...
1  import json
2  import boto3
3  import os
4  from datetime import datetime
5
6  def lambda_handler(event, context):
7      # Get current execution count from environment variable
8      execution_count = int(os.environ.get('EXECUTION_COUNT', 0))
9
10     # Increment execution count
11     execution_count += 1
12
13     # Update execution count in environment variable
14     os.environ['EXECUTION_COUNT'] = str(execution_count)
15
16     # Stop execution after 3 runs
17     if execution_count > 3:
18         return {
19             'statusCode': 200,
20             'body': 'Execution stopped after 3 runs'
21         }
22
23     # Generate JSON data
24     json_data = {
25         "transaction_id": 12345,
26         "payment_mode": "card/netbanking/upi",
27         "amount": 200.0,
28         "customer_id": 101,
29         "timestamp": str(datetime.now())
30     }
31
32     # Save JSON data to S3
33     bucket_name = 'sonam25ranchibucket'
34     file_name = f"transaction_{datetime.now().strftime('%Y-%m-%d_%H-%M-%S')}.json"
35
36     s3 = boto3.client('s3')
37     s3.put_object(Bucket=bucket_name, Key=file_name, Body=json.dumps(json_data))
38
39     return {
40         'statusCode': 200,
41         'body': json.dumps('JSON file saved successfully')
42     }
43
44
```

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 file:///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-fu
nction"

```
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

The screenshot shows the Amazon S3 console interface. At the top, it displays the path: Amazon S3 > Buckets > sonam25ranchibucket. Below this, the bucket name 'sonam25ranchibucket' is shown with a 'Info' link. A navigation bar below the bucket name includes tabs for Objects, Properties, Permissions, Metrics, Management, and Access Points. The 'Objects' tab is selected. In the main content area, there is a heading 'Objects (3) Info' with a 'Actions' dropdown menu and a 'Create folder' and 'Upload' buttons. Below this, a search bar says 'Find objects by prefix'. A table lists three objects:

	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	transaction_2024-05-03_10-32-36.json	json	May 3, 2024, 16:02:39 (UTC+05:30)	144.0 B	Standard
<input type="checkbox"/>	transaction_2024-05-03_10-33-19.json	json	May 3, 2024, 16:03:20 (UTC+05:30)	144.0 B	Standard
<input type="checkbox"/>	transaction_2024-05-03_10-33-52.json	json	May 3, 2024, 16:03:53 (UTC+05:30)	144.0 B	Standard

At the bottom of the page, there are links for CloudShell, Feedback, © 2024, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

logs>>

The screenshot shows the AWS CloudWatch Log Group details page for the log group '/aws/lambda/my-lambda-function'. The top navigation bar includes 'Actions', 'View in Logs Insights', 'Start tailing', and 'Search log group'. Below the header, there's a section titled 'Log group details' with various configuration settings:

Log class	Info	Metric filters	Anomaly detection
Standard	0	Configure	
ARN	arn:aws:logs:us-east-1:058264126755:log-group:/aws/lambda/my-lambda-function*	Subscription filters	Data protection
Creation time	1 hour ago	0	-
Retention	Never expire	Contributor Insights rules	Sensitive data count
Stored bytes	-	KMS key ID	-

Below this, there are tabs for 'Log streams', 'Tags', 'Anomaly detection', 'Metric filters', 'Subscription filters', 'Contributor Insights', and 'Data protection'. The 'Log streams' tab is selected, showing five log streams with their ARNs and last event times:

Log stream	Last event time
2024/05/03/[\$LATEST]4ec864964f8e49d1979d3da0789a774a	2024-05-03 16:02:38 (UTC+05:30)
2024/05/03/[\$LATEST]ada8701872ca45dda1deb01ae22c9d99	2024-05-03 15:31:54 (UTC+05:30)
2024/05/03/[\$LATEST]161a21e6b8b540f9868a95eea3f5c101	2024-05-03 15:17:24 (UTC+05:30)

At the bottom right, there are links for 'Privacy', 'Terms', and 'Cookie preferences'.

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})
}
```

Successfully updated the function my-lambda-function.

Code source [Info](#)

File Edit Find View Go Tools Window Test Deploy

Environment Var lambda_function Execution results

```

1 import json
2 import boto3
3 from datetime import datetime
4
5 def lambda_handler(event, context):
6     # Get parameters from the request body
7     transaction_id = event.get('transaction_id')
8     payment_mode = event.get('payment_mode')
9     amount = event.get('amount')
10    customer_id = event.get('customer_id')
11
12    # Check if required parameters are present
13    if not all([transaction_id, payment_mode, amount, customer_id]):
14        return {
15            'statusCode': 400,
16            'body': json.dumps({'error': 'Missing required parameters'})
17        }
18
19    # Generate JSON data with current timestamp
20    timestamp = datetime.now().isoformat()
21    json_data = {
22        "transaction_id": transaction_id,
23        "payment_mode": payment_mode,
24        "amount": amount,
25        "customer_id": customer_id,
26        "timestamp": timestamp
27    }
28
29    # Save JSON data to S3
30    bucket_name = 'sonam25branchbucket'
31    file_name = f"transaction_{timestamp}.json"
32
33    s3 = boto3.client('s3')

```

30:40 Python Spaces: 4

Code properties [Info](#)

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 --region 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 --authorization-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 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
{
  "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:MyLambdaFunction/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": "m06751",
  "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

The screenshot shows the AWS S3 console interface. The top navigation bar includes 'Amazon S3 > Buckets > sonam25ranchibucket'. Below the navigation is a header with tabs: 'Objects' (selected), 'Properties', 'Permissions', 'Metrics', 'Management', and 'Access Points'. The main area is titled 'Objects (8) Info' with a 'Upload' button. A sub-header indicates 'Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions.' Below this is a search bar labeled 'Find objects by prefix'. The main table lists eight objects, each with a checkbox, name, type (json), last modified date, size, and storage class (Standard). The objects are named transaction_2024-05-03T12:50:46.773398.json through transaction_2024-05-03T12:53:54.932058.json.

	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	transaction_2024-05-03T12:50:46.773398.json	json	May 3, 2024, 18:20:49 (UTC+05:30)	144.0 B	Standard
<input type="checkbox"/>	transaction_2024-05-03T12:59:00.242118.json	json	May 3, 2024, 18:09:01 (UTC+05:30)	126.0 B	Standard
<input type="checkbox"/>	transaction_2024-05-03T12:37:36.067933.json	json	May 3, 2024, 18:07:39 (UTC+05:30)	126.0 B	Standard
<input type="checkbox"/>	transaction_2024-05-03T12:29:34.288261.json	json	May 3, 2024, 17:59:37 (UTC+05:30)	126.0 B	Standard
<input type="checkbox"/>	transaction_2024-05-03T10:53:54.932058.json	json	May 3, 2024, 16:23:58 (UTC+05:30)	144.0 B	Standard
<input type="checkbox"/>	transaction_2024-05-03-10-33-52.json	json	May 3, 2024, 16:03:53 (UTC+05:30)	144.0 B	Standard
<input type="checkbox"/>	transaction_2024-05-03-10-33-19.json	json	May 3, 2024, 16:03:20 (UTC+05:30)	144.0 B	Standard
<input type="checkbox"/>	transaction_2024-05-03-10-32-36.json	json	May 3, 2024, 16:02:39 (UTC+05:30)	144.0 B	Standard