Create S3 bucket from AWS CLI

- Configure aws with the terminal
  Pip3 install aws
  Pip3 install awscli

```
~ ?                                                    ✔ 8s 12:39:37
└─ aws configure
AWS Access Key ID [None]: AKIAZI2LHWBXGJ67XRXA
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json
```

- Create iam role:

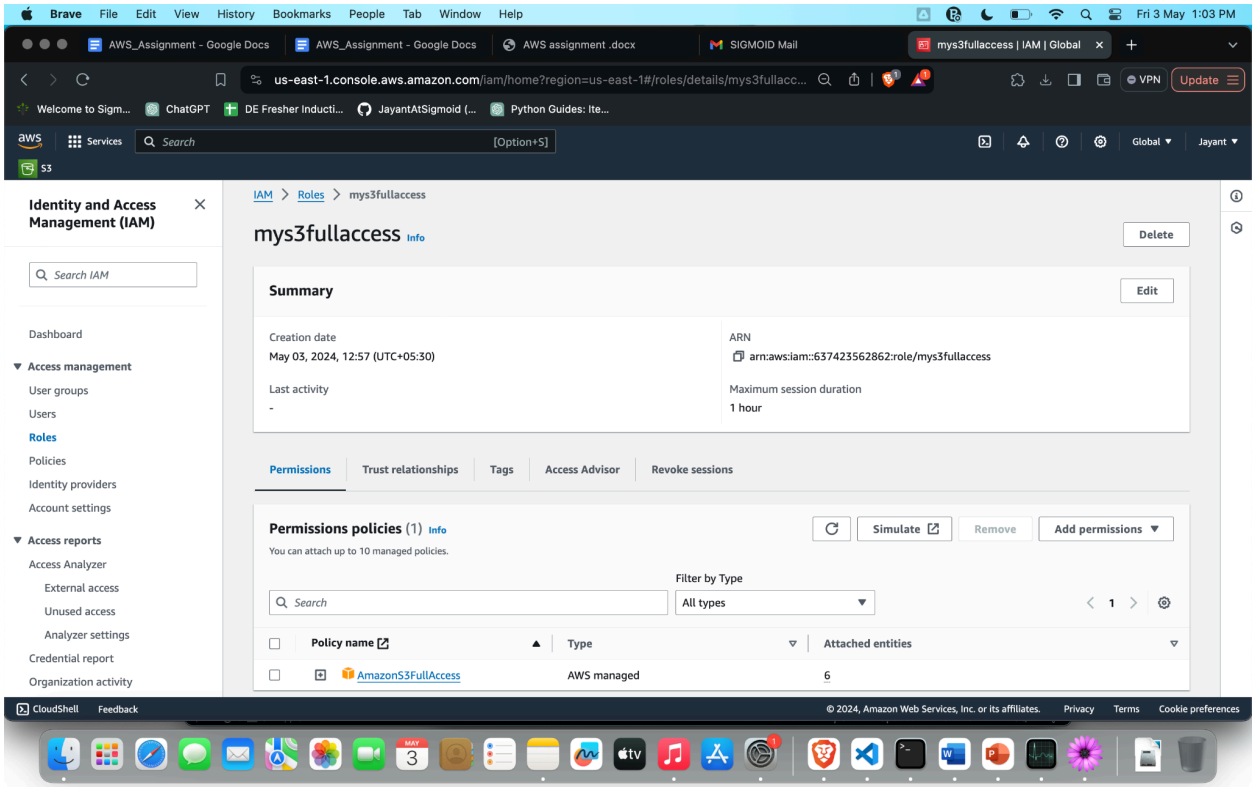
```
~ ?                                                    ? 255 ✘ 12:55:35
└─ aws iam create-role --role-name mys3fullaccess --assume-role-policy-document
file:///Users/jayantasudhani/Documents/AWS/AWS_Assignment/trust_policy.json
{
    "Role": {
        "Path": "/",
        "RoleName": "mys3fullaccess",
        "RoleId": "AROAZI2LHWBXPB4T5HNMR",
        "Arn": "arn:aws:iam::637423562862:role/mys3fullaccess",
        "CreateDate": "2024-05-03T07:27:11Z",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "lambda.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }
    }
```

Trust-policy.json

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

- Attach S3 Full Access policy to the above created role:

```
aws iam attach-role-policy --role-name mys3fullaccess --policy-arn arn:aws:iam::aws:policy/AmazonS3FullAccess
```



- Create an IAM instance profile and attach it to the above role:

```
~ ✔ 13:06:08
aws iam add-role-to-instance-profile --role-name mys3fullaccess --instance-profile-name MyInstanceProfile
```

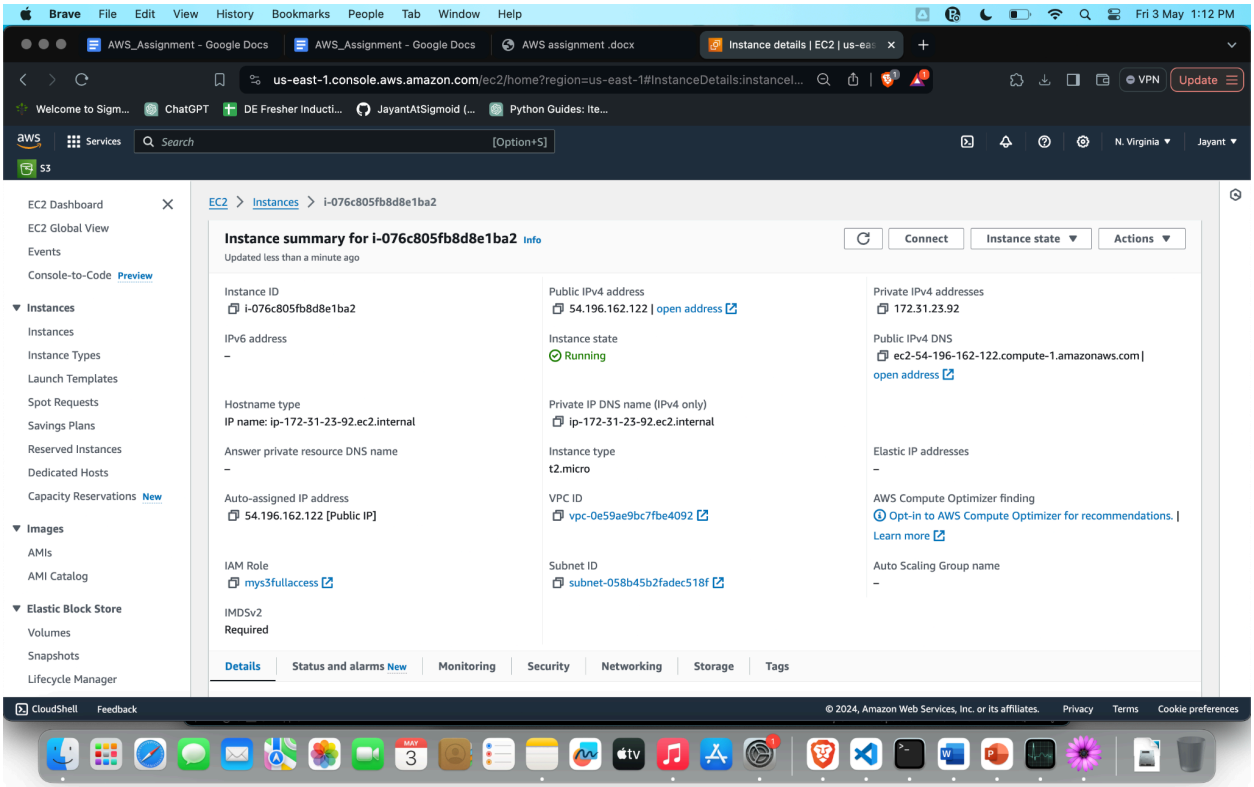
```
~ ✔ 13:00:58
aws iam create-instance-profile --instance-profile-name MyInstanceProfile

{
    "InstanceProfile": {
        "Path": "/",
        "InstanceProfileName": "MyInstanceProfile",
        "InstanceProfileId": "AIPAZI2LHWBXHY2FFNPUL",
        "Arn": "arn:aws:iam::637423562862:instance-profile/MyInstanceProfile",
        "CreateDate": "2024-05-03T07:36:08Z",
        "Roles": []
    }
}
```

- List IAM instance profiles:

```
~ ✔ 13:06:54
aws iam list-instance-profiles

{
    "InstanceProfiles": [
        {
            "Path": "/",
            "InstanceProfileName": "AmazonEMR-InstanceProfile-20240417T154613",
            "InstanceProfileId": "AIPAZI2LHWBXCS2YIMAWO",
            "Arn": "arn:aws:iam::637423562862:instance-profile/AmazonEMR-InstanceProfile-20240417T154613",
            "CreateDate": "2024-04-17T10:16:16Z",
            "Roles": [
                {
                    "Path": "/service-role/",
                    "RoleName": "AmazonEMR-InstanceProfile-20240417T154613",
                    "RoleId": "AROAZI2LHWBXBCMPUEQME",
                    "Arn": "arn:aws:iam::637423562862:role/service-role/AmazonEMR-InstanceProfile-20240417T154613",
                    "CreateDate": "2024-04-17T10:16:15Z",
                    "AssumeRolePolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {
                                    "Service": "ec2.amazonaws.com"
                                },
                                "Action": "sts:AssumeRole"
                            }
                        ]
                    }
                }
            ]
        },
        {
            "Path": "/",
            "InstanceProfileName": "AmazonEMR-InstanceProfile-20240418T121701",
            "InstanceProfileId": "AIPAZI2LHWBXPC52AWH5F",
            "Arn": "arn:aws:iam::637423562862:instance-profile/AmazonEMR-InstanceProfile-20240418T121701",
```

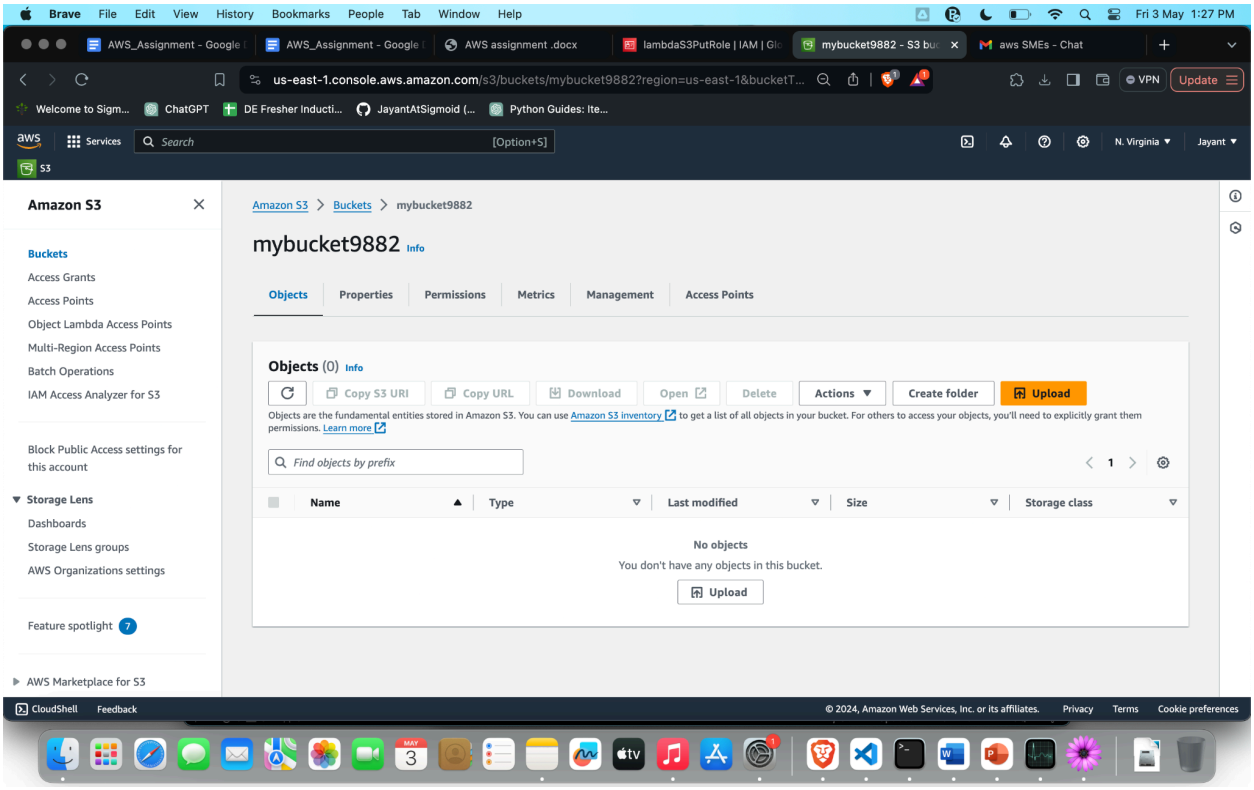
- Create an EC2 instance using above role and key:

```
13:07:45
└─ aws ec2 run-instances --image-id ami-07caf09b362be10b8 --count 1 --instance-type t2.micro --key-name key_assignment --iam-insta
nce-profile Name=MyInstanceProfile --region us-east-1
{
    "Groups": [],
    "Instances": [
        {
            "AmiLaunchIndex": 0,
            "ImageId": "ami-07caf09b362be10b8",
            "InstanceId": "i-076c805fb8d8e1ba2",
            "InstanceType": "t2.micro",
            "KeyName": "key_assignment",
            "LaunchTime": "2024-05-03T07:41:23.000Z",
            "Monitoring": {
                "State": "disabled"
            },
            "Placement": {
                "AvailabilityZone": "us-east-1a",
                "GroupName": "",
                "Tenancy": "default"
            },
            "PrivateDnsName": "ip-172-31-23-92.ec2.internal",
            "PrivateIpAddress": "172.31.23.92",
            "ProductCodes": [],
            "PublicDnsName": "",
            "State": {
                "Code": 0,
                "Name": "pending"
            },
            "StateTransitionReason": "",
            "SubnetId": "subnet-058b45b2fadec518f",
            "VpcId": "vpc-0e59ae9bc7fbe4092",
            "Architecture": "x86_64",
            "BlockDeviceMappings": [],
            "ClientToken": "ea405ad0-f12e-4e71-b2a2-ea54c7ceb026",
            "EbsOptimized": false,
            "EnaSupport": true,
```



- Create s3 bucket:

```
aws s3api create-bucket --bucket mybucket9882 --region us-east-1

{
    "Location": "/mybucket9882"
}
```



Put files in S3 bucket from lambda

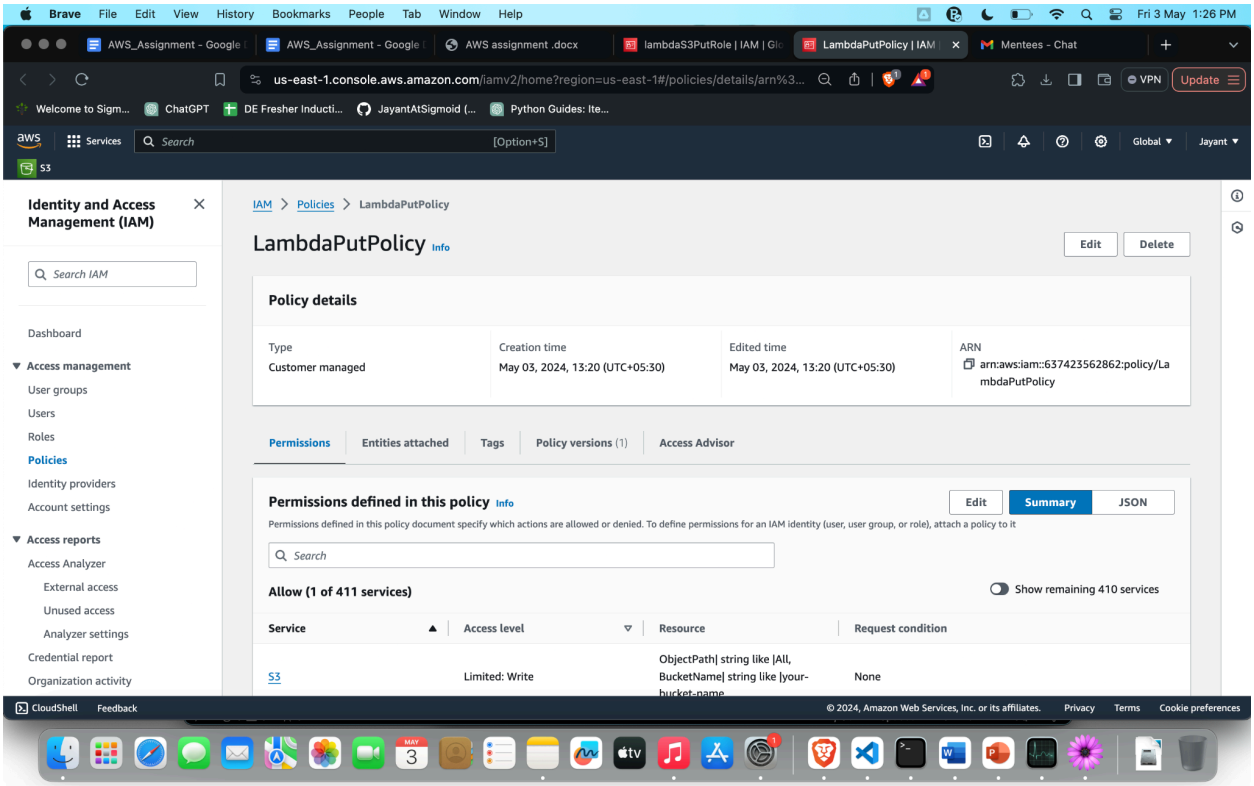
- Create policy to put object in S3
  putObjectPolicy.json

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::vrabuckvra/*"
        }
    ]
}
```

```
~ ?                                                                                                    ✔ ? 13:13:03
└─ aws iam create-policy --policy-name LambdaPutPolicy --policy-document file:///Users/jayantasudhani/Documents/AWS/AWS_Assignment
/s3putobjectpolicy.json
{
    "Policy": {
        "PolicyName": "LambdaPutPolicy",
        "PolicyId": "ANPAZI2LHWBXE3EFRGWHZ",
        "Arn": "arn:aws:iam::637423562862:policy/LambdaPutPolicy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2024-05-03T07:50:47Z",
        "UpdateDate": "2024-05-03T07:50:47Z"
    }
}
```

- Create a new IAM role and attach the above policy:

```
~ ?                                                                                                    ✔ ? 13:20:47
└─ aws iam create-role --role-name lambdaS3PutRole --assume-role-policy-document file:///Users/jayantasudhani/Documents/AWS/AWS_As
signment/trust_policy.json
{
    "Role": {
        "Path": "/",
        "RoleName": "lambdaS3PutRole",
        "RoleId": "AROAZI2LHWBXLRJCQTYYD",
        "Arn": "arn:aws:iam::637423562862:role/lambdaS3PutRole",
        "CreateDate": "2024-05-03T07:53:34Z",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "lambda.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }
    }
}
```



- Create an IAM policy that allows logging to CloudWatch Logs and attach to the above created role:
  cloudWatchPolicies.json

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

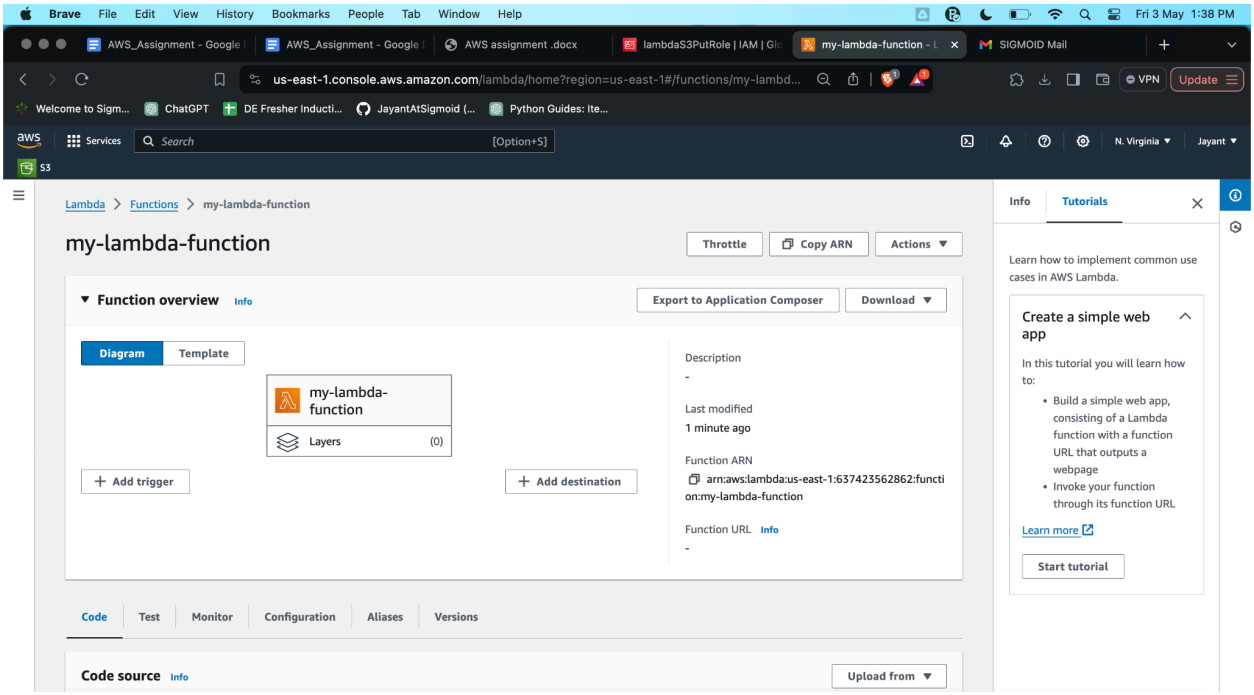
```
aws iam create-policy --policy-name CloudWatchPolicy --policy-document file:///Users/jayantasudhani/Documents/AWS/AWS_Assignment/cloudWatchPolicies.json
{
    "Policy": {
        "PolicyName": "CloudWatchPolicy",
        "PolicyId": "ANPAZI2LHWBXBAIXAPMB4",
        "Arn": "arn:aws:iam::637423562862:policy/CloudWatchPolicy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2024-05-03T08:01:54Z",
        "UpdateDate": "2024-05-03T08:01:54Z"
    }
}
```

- Create lambda function to put file in the bucket
  Lambda_function.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 = 'vrabuckvra'
    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')
    }
```

```
~ ?                                                                    13:33:01
└─ aws lambda create-function \
            --function-name my-lambda-function \
    --runtime python3.8 \
            --role arn:aws:iam::637423562862:role/lambdaS3PutRole \
            --handler lambda_function.lambda_handler \
            --zip-file fileb:///Users/jayantasudhani/Documents/AWS/AWS_Assignment/Lambda_function.py.zip

{
    "FunctionName": "my-lambda-function",
    "FunctionArn": "arn:aws:lambda:us-east-1:637423562862:function:my-lambda-function",
    "Runtime": "python3.8",
    "Role": "arn:aws:iam::637423562862:role/lambdaS3PutRole",
    "Handler": "lambda_function.lambda_handler",
    "CodeSize": 1130,
    "Description": "",
    "Timeout": 3,
    "MemorySize": 128,
    "LastModified": "2024-05-03T08:06:53.841+0000",
    "CodeSha256": "R6lWiBt8bjyPTg3TqEUmk5/NIHA/zMW3ZvpscxogG1k=",
    "Version": "$LATEST",
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "RevisionId": "ec946fb5-716f-4f00-8af3-885163078f22",
    "State": "Pending",
    "StateReason": "The function is being created.",
    "StateReasonCode": "Creating",
    "PackageType": "Zip",
    "Architectures": [
        "x86_64"
    ],
    "EphemeralStorage": {
        "Size": 512
    },
    "SnapStart": {
```



- Create event to trigger the lambda_function every minute for 3 times

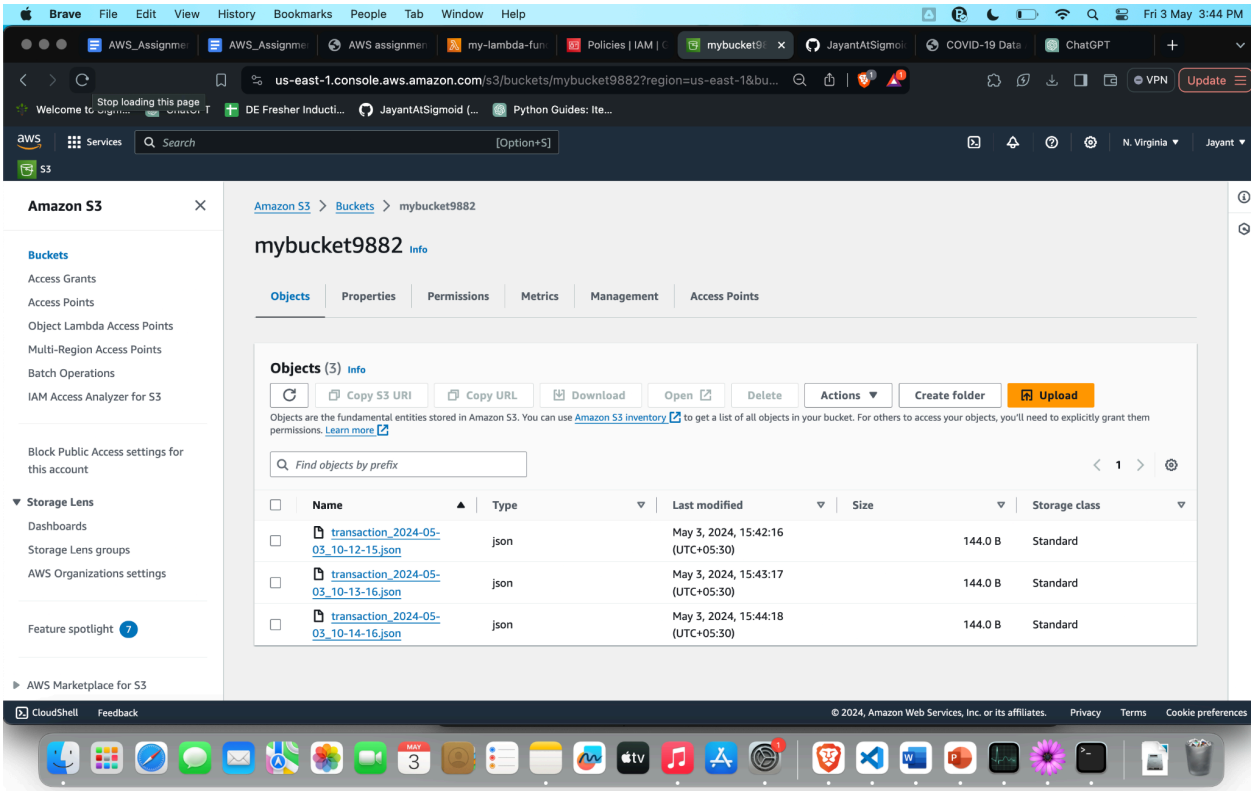
  Create event rule:

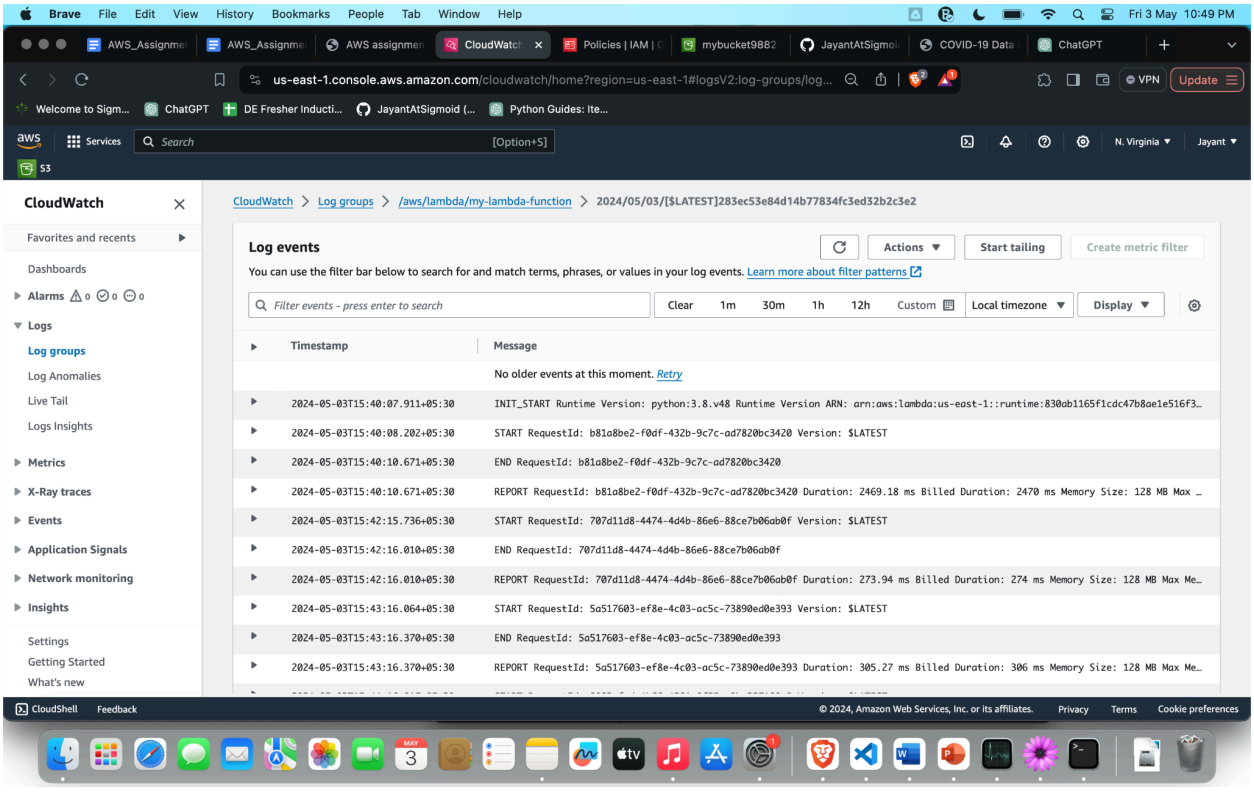
```
aws events put-rule --name "LambdaExecutionScheduleRule" \
    --schedule-expression "rate(1 minute)" \
    --state "ENABLED" \
    --description "Trigger Lambda function every minute" \
    --event-pattern "{\"source\":[\"aws.events\"],\"detail-type\":[\"Scheduled Event\"],\"resources\":[\"arn:aws:events:us-east-1:637423562862:rule/LambdaExecutionScheduleRule\"]}" \



{
    "RuleArn": "arn:aws:events:us-east-1:637423562862:rule/LambdaExecutionScheduleRule"
}
```

  Add rule to the function:

```
aws events put-targets --rule "LambdaExecutionScheduleRule" --targets "Id"="1","Arn"="arn:aws:lambda:us-east-1:637423562862:function:my-lambda-function"

{
    "FailedEntryCount": 0,
    "FailedEntries": []
}
```

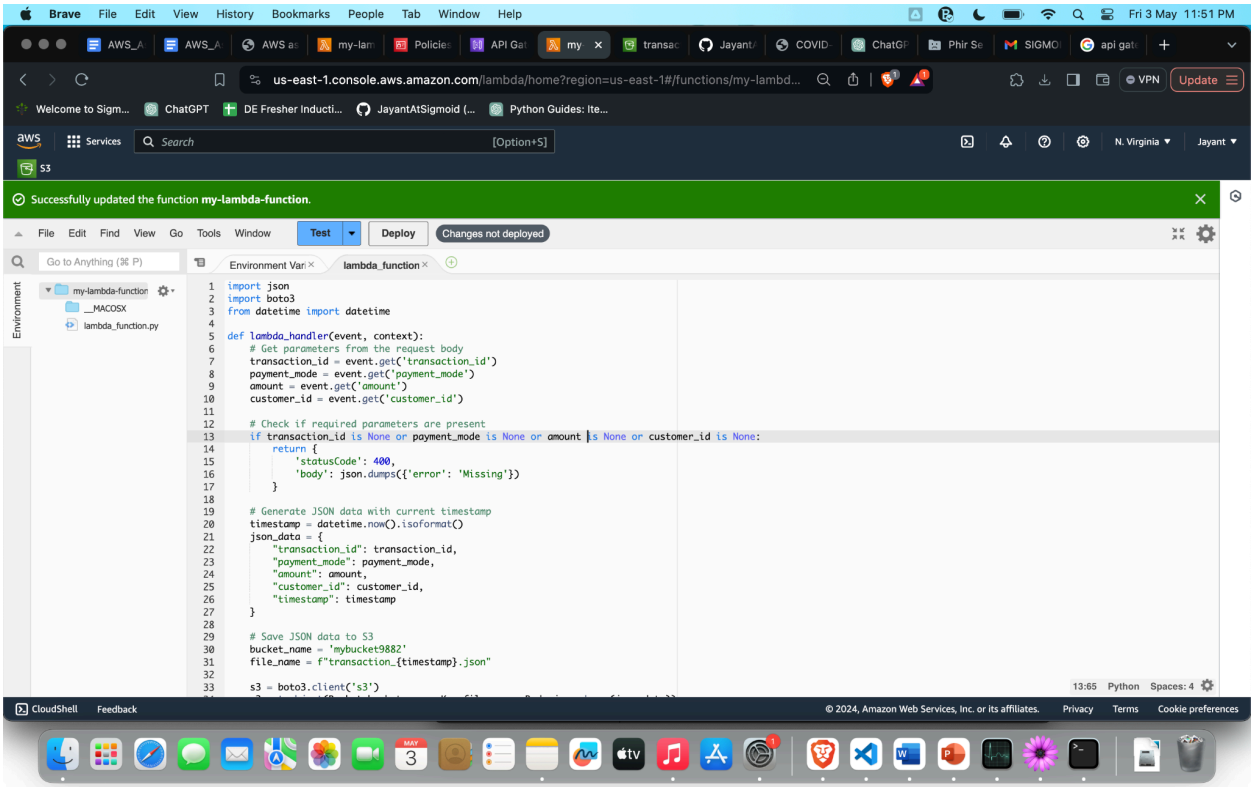
- On execution of the lambda_function filles got loaded in S3 bucket every minute for 3 times





API gateway - Lambda integration

- Modify the lambda function to accept parameters and return file name. My_lambda_function.py:

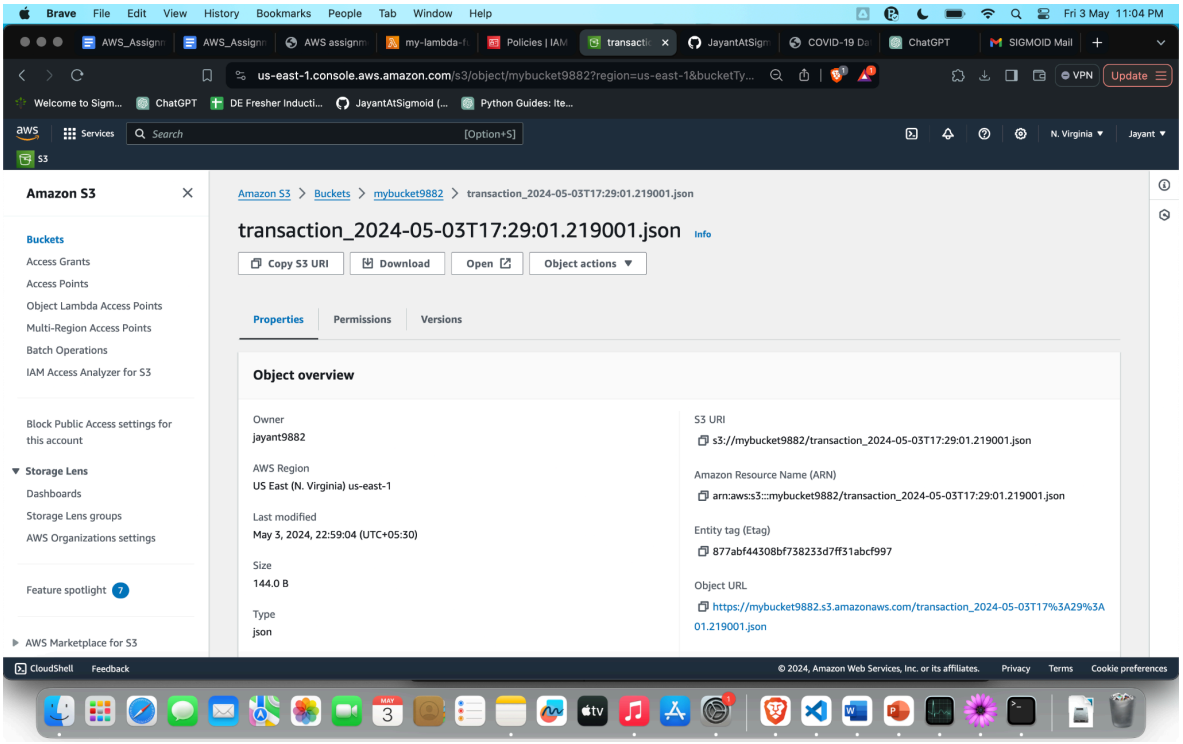



```python
import json
import boto3
from datetime import datetime

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 transaction_id is None or payment_mode is None or amount is None or customer_id is None:
```

```
        return {
            'statusCode': 400,
            'body': json.dumps({'error': 'Missing'})
        }

    # 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 = 'mybucket9882'
    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})
    }
```

Deploying and invoking this function

```
~ 22:57:18
└─ aws lambda invoke --function-name my-lambda-function  --payload '{"transaction_id": 12345, "payment_mode": "card/netbanking/upi
", "amount": 200.0,"customer_id": 101}' output.json
{
    "StatusCode": 200,
    "ExecutedVersion": "$LATEST"
}
```



On checking this function independently, file successfully uploaded in bucket.

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

Creating rest api from api gateway

```
aws apigateway create-rest-api --name 'FileAPI' --region us-east-1
{
    "id": "iqjl6tod3j",
    "name": "FileAPI",
    "createdDate": 1714757999,
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
        "types": [
            "EDGE"
        ]
    },
    "disableExecuteApiEndpoint": false,
    "rootResourceId": "6jf7k9jm8g"
}
```

```
aws apigateway get-resources --rest-api-id iqjl6tod3j --region us-east-1
{
    "items": [
        {
            "id": "6jf7k9jm8g",
            "path": "/"
        }
    ]
}
```

```
~ ?                                                                    ?255 ✘ ? 23:11:01
└─ aws apigateway create-resource --rest-api-id iqjl6tod3j --parent-id 6jf7k9jm8g --path-part files --region us-east-1
{
    "id": "0cus3y",
    "parentId": "6jf7k9jm8g",
    "pathPart": "files",
    "path": "/files"
}
```

Assigning POST request.

```
~ ?                                                                    ✔ ? 23:11:22
└─ aws apigateway put-method --rest-api-id iqjl6tod3j --resource-id 6jf7k9jm8g --http-method POST --authorization-type NONE --regi
on us-east-1
{
    "httpMethod": "POST",
    "authorizationType": "NONE",
    "apiKeyRequired": false
}
```

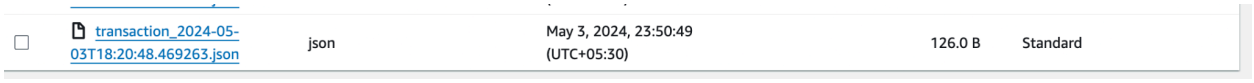
```
~ ?                                                                    ✔ ? 23:12:09
└─ aws apigateway put-integration --rest-api-id iqjl6tod3j --resource-id 6jf7k9jm8g --http-method POST --type AWS_PROXY --integrat
ion-http-method POST --uri 'arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-east-1:637423562862:fu
nction:my-lamda-function/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-1:637423562862:function:my-lamda-
function/invocations",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "timeoutInMillis": 29000,
    "cacheNamespace": "6jf7k9jm8g",
    "cacheKeyParameters": []
}
```

Deploying in prod

```
~ ?                                                                    ✔ ? 23:13:49
└─ aws apigateway create-deployment --rest-api-id iqjl6tod3j --stage-name prod --region us-east-1
{
    "id": "7h4oqd",
    "createdDate": 1714758278
}
```

c. Consume API from local machine and pass unique data to lambda. Using curl to check the api

```
~ ?                                                                    ✔ ? 23:44:27
└─ curl -X POST  'https://iqjl6tod3j.execute-api.us-east-1.amazonaws.com/prod/my-lambda-function' \
-d '{"operation": "create", "payload": {"Item":{ "transaction_id": 12345, "payment_mode": "card/netbanking/upi", "amount": 200.0,
"customer_id": 101, "timestamp":"77" }}'
{"file_name": "transaction_2024-05-03T18:20:48.469263.json"}%
```



d. Check if cloudwatch logs are generated Cloudwatch logs: