\\ Hosting a html form in S3 bucket and get the form data store it in the same S3 bucket using lambda and API gateway //

Here are the steps:

1. Create an S3 bucket: Log in to the AWS Management Console and create an S3 bucket in the desired region.
2. Create an HTML form: Create an HTML form that will allow users to enter data that you want to store in the S3 bucket.
3. Upload the HTML form to the S3 bucket: Upload the HTML form to the S3 bucket you created in step 1.
4. Create an API Gateway: Create an API Gateway that will be used to capture data submitted by the HTML form.
5. Create a Lambda function: Create a Lambda function that will be used to process data submitted by the API Gateway and store it in the S3 bucket.
6. Configure API Gateway to integrate with Lambda: Configure the API Gateway to integrate with the Lambda function you created in step 5.
7. Add CORS configuration to the S3 bucket: Add a CORS (Cross-Origin Resource Sharing) configuration to the S3 bucket to allow the HTML form to make requests to the API Gateway.
8. Test the integration: Test the integration by submitting data through the HTML form and verifying that the data is stored in the S3 bucket.

Here is a more detailed step-by-step guide:

Step 1: Create an S3 bucket

1. Log in to the AWS Management Console.
2. Navigate to the S3 service.
3. Click on the "Create Bucket" button.
4. Enter a unique name for the bucket.
5. Select the desired region for the bucket.
6. Leave the default settings for the other options and click on the "Create Bucket" button.

Step 2: Create an HTML form

1. Create an HTML form that will allow users to enter data that you want to store in the S3 bucket.
2. The HTML form should have input fields for the data you want to capture.
3. When the form is submitted, it should send a POST request to the API Gateway.

Step 3: Upload the HTML form to the S3 bucket

1. Navigate to the S3 bucket you created in step 1.
2. Click on the "Upload" button.
3. Upload the HTML form you created in step 2 to the bucket.

Step 4: Create an API Gateway

1. Navigate to the API Gateway service in the AWS Management Console.
2. Click on the "Create API" button.
3. Select the "REST API" option.
4. Select the "New API" option.
5. Enter a name for the API and click on the "Create API" button.

Step 5: Create a Lambda function

1. Navigate to the Lambda service in the AWS Management Console.
2. Click on the "Create Function" button.
3. Select the "Author from scratch" option.
4. Enter a name for the function.
5. Select the runtime as "Python".
6. Leave the default settings for the other options and click on the "Create Function" button.
7. Write code to process the data submitted by the API Gateway and store it in the S3 bucket.

Here is a sample Python code to store data in S3:

import boto3

s3 = boto3.client('s3')

def lambda\_handler(event, context):

data = event['body']

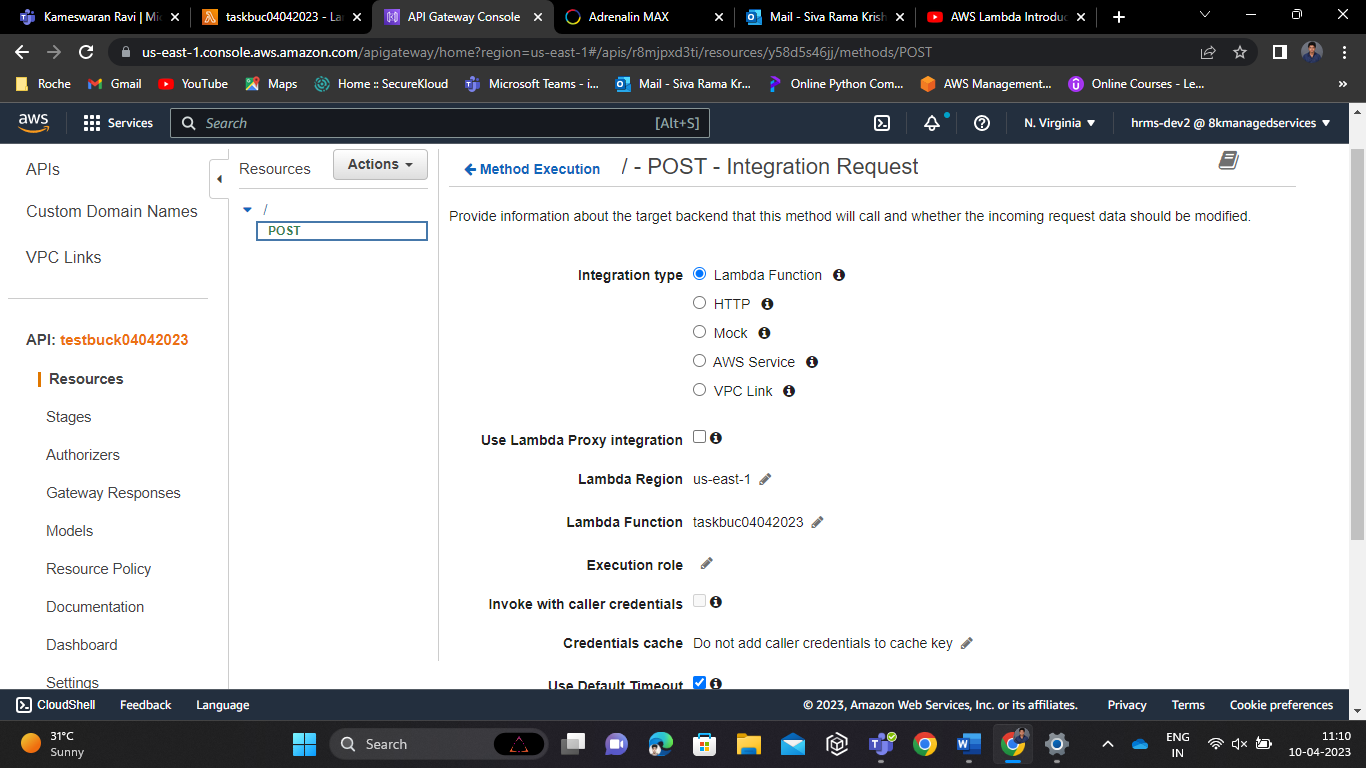
s3.put\_object(Bucket='bucket-name', Key='file-name', Body=data)

Step 6: Configure API Gateway to integrate with Lambda

1. Navigate to the API Gateway you created in step 4.
2. Click on the "Create Resource" button
3. Enter a name for the resource and click on the "Create Resource" button.
4. Click on the "Create Method" button and select "POST".
5. Select "Lambda Function" as the integration type.
6. Select the region where your Lambda function is located.
7. Enter the name of the Lambda function you created in step 5.
8. Click on the "Save" button.
9. Click on the "Actions" button and select "Deploy API".
10. Select the desired deployment stage and click on the "Deploy" button.

Create a S3 bucket, create a index.html file and stored it in the same S3 bucket and give the access for index.html file as public and enable the web hosting for the same S3 bucket. now create the lambda for this task.and edit the permission through the lambda and modify the role with S3 full access and API gateway invoke full access for that go to the configuration tab in lambda dashboard give there permissions.

Then create a API gateway and create post method



With that post method, We are integrated the lambda function with that API gateway.After that Deploy the API.

import boto3

import json

s3 = boto3.client('s3')

def lambda\_handler(event, context):

# Get the submitted form data from the S3 bucket

bucket\_name = 'taskbuck04042023'

form\_key = 'index.html'

form\_data = s3.get\_object(Bucket=bucket\_name, Key=form\_key)['Body'].read().decode('utf-8')

# Process the form data (e.g., extract form fields, validate input, etc.)

submitted\_data = process\_form\_data(event)

# Check if folders already exist, if not create them

folder\_prefix = 'sample'

response = s3.list\_buckets()

if bucket\_name not in [bucket['Name'] for bucket in response ['Buckets']]:

s3.create\_bucket(Bucket=bucket\_name)

# Get the current folder number and file number from S3 or set them to 1 and 0 respectively if they don't exist

folder\_number\_key = 'folder\_number.txt'

try:

folder\_number = int(s3.get\_object(Bucket=bucket\_name, Key=folder\_number\_key)['Body'].read().decode('utf-8'))

except s3.exceptions.NoSuchKey:

folder\_number = 1

file\_number\_key = 'file\_number.txt'

try:

file\_number = int(s3.get\_object(Bucket=bucket\_name, Key=file\_number\_key)['Body'].read().decode('utf-8'))

except s3.exceptions.NoSuchKey:

file\_number = 0

# Create a new folder if the current folder is full (i.e., it has 10 files)

if file\_number % 4 == 0:

current\_folder = f"{folder\_prefix}{folder\_number}/"

s3.put\_object(Bucket=bucket\_name, Key=current\_folder, Body='')

folder\_number += 1

s3.put\_object(Bucket=bucket\_name, Key=folder\_number\_key, Body=bytes(str(folder\_number), 'utf-8'))

else:

current\_folder = f"{folder\_prefix}{folder\_number - 1}/"

# Put the processed data into a new S3 file with a sequential number in the file name

result\_key = f'{current\_folder}result\_{file\_number}.txt'

s3.put\_object(Bucket=bucket\_name, Key=result\_key, Body=bytes(submitted\_data, 'utf-8'))

# Increment the file number counter and save it back to S3

file\_number += 1

s3.put\_object(Bucket=bucket\_name, Key=file\_number\_key, Body=bytes(str(file\_number), 'utf-8'))

# Return a response to the user (e.g., a success message)

response = {

'statusCode': 200,

'body': 'Form submitted successfully!'

}

return response

def process\_form\_data(body):

# Parse the form data and extract the fields

# form\_data = json.loads(body)

form\_data=body

name = form\_data['Name']

email = form\_data['Email']

message = form\_data['Message']

# Validate the input (e.g., ensure fields are not empty)

if not name or not email or not message:

raise ValueError('All form fields are required')

# Construct the processed data and return it

processed\_data = {

'Name': name,

'Email': email,

'Message': message

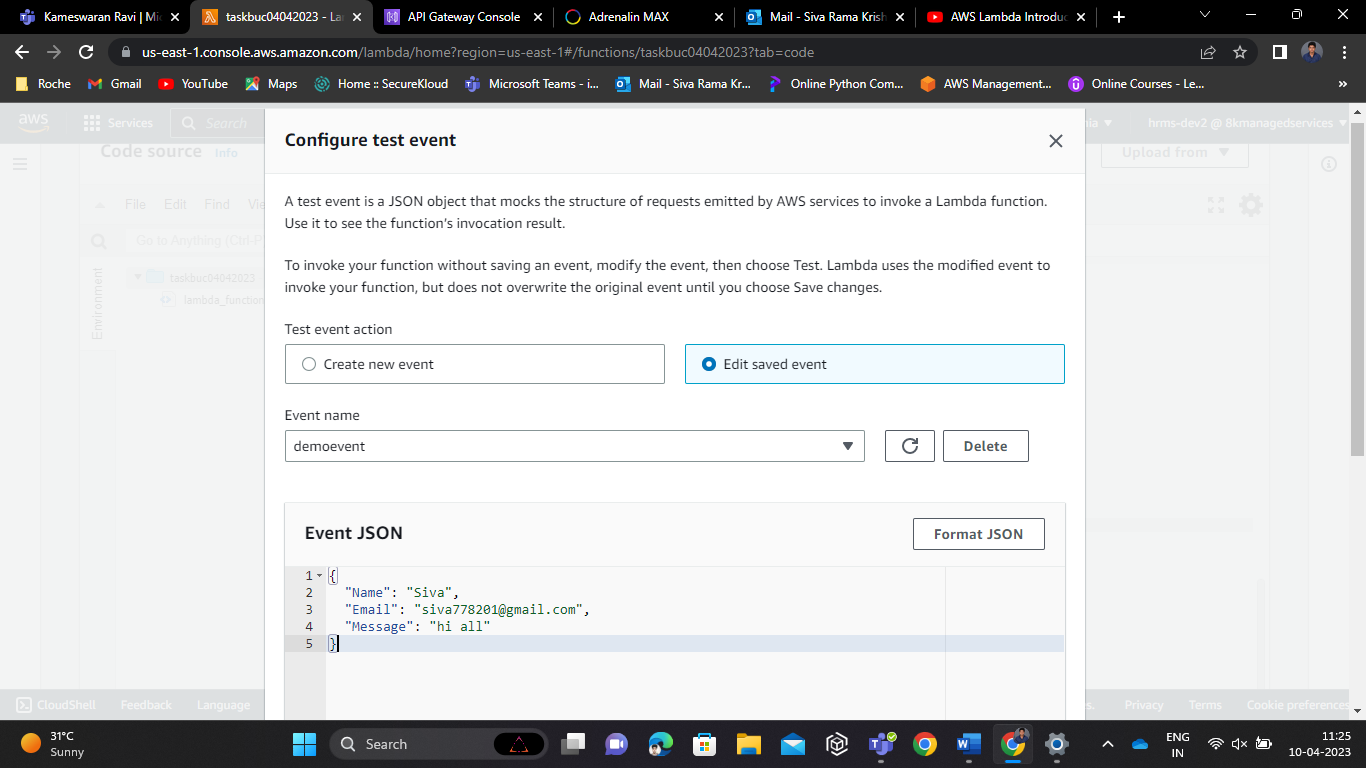
}

print(processed\_data)

return json.dumps(processed\_data)

By using this code, We can create a new folder .each folder have a ten input when we cross the ten inputs(form data),a new folder gets automatically created there.

For this we have to configure the test event, based on our html structure.



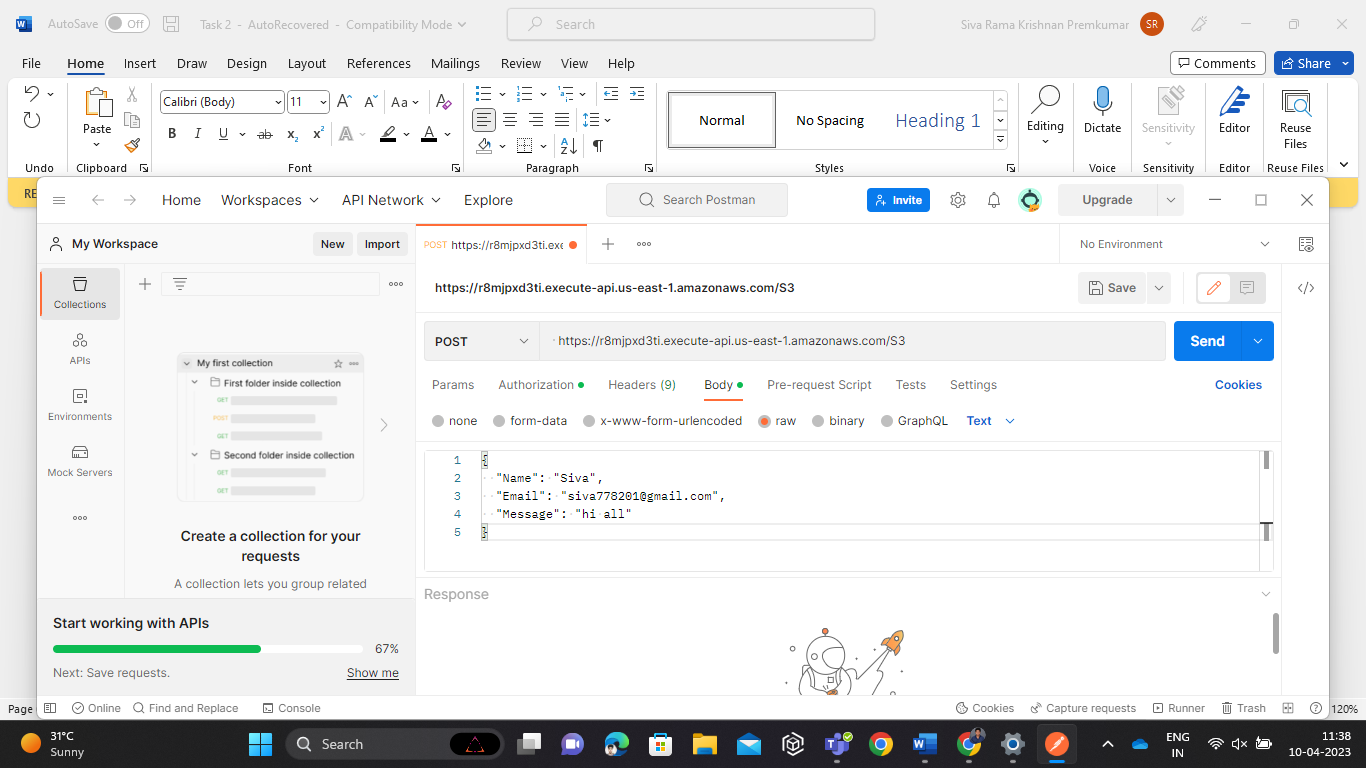
After this whenever we test this event ,we got the result like this, below mentioned..

A screenshot of a computer

Description automatically generated

So when we test this configuration and its show success code for this kind of every action there is a form data created and stored in the S3 bucket folder automatically.

If we need to show its working fine through the API gateway, Then copy and paste the API gateway invoke URL in postman application, change the method into post then touch the send button.. if it doesn’t works well.. bypass the key value like this.. below mentioned..



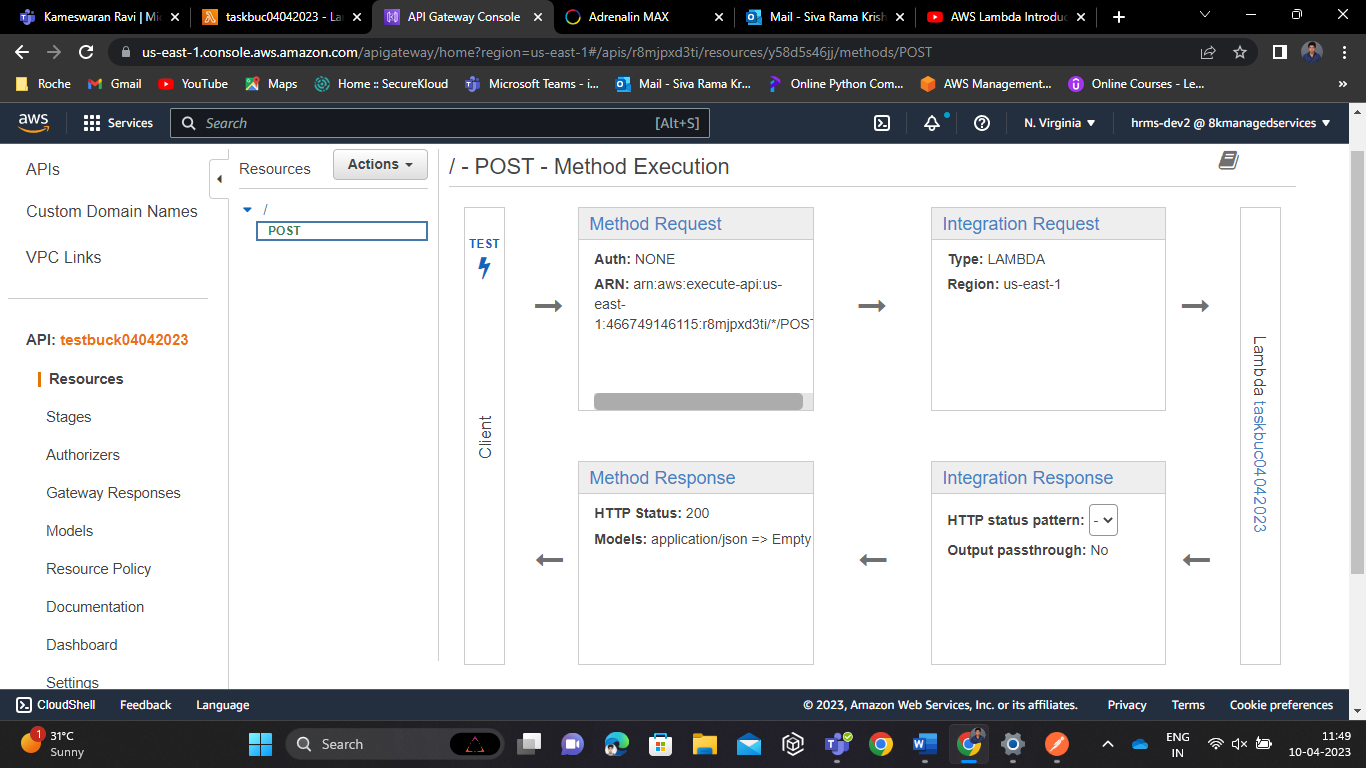
After this we got a success code.

How to authenticate API gatway:

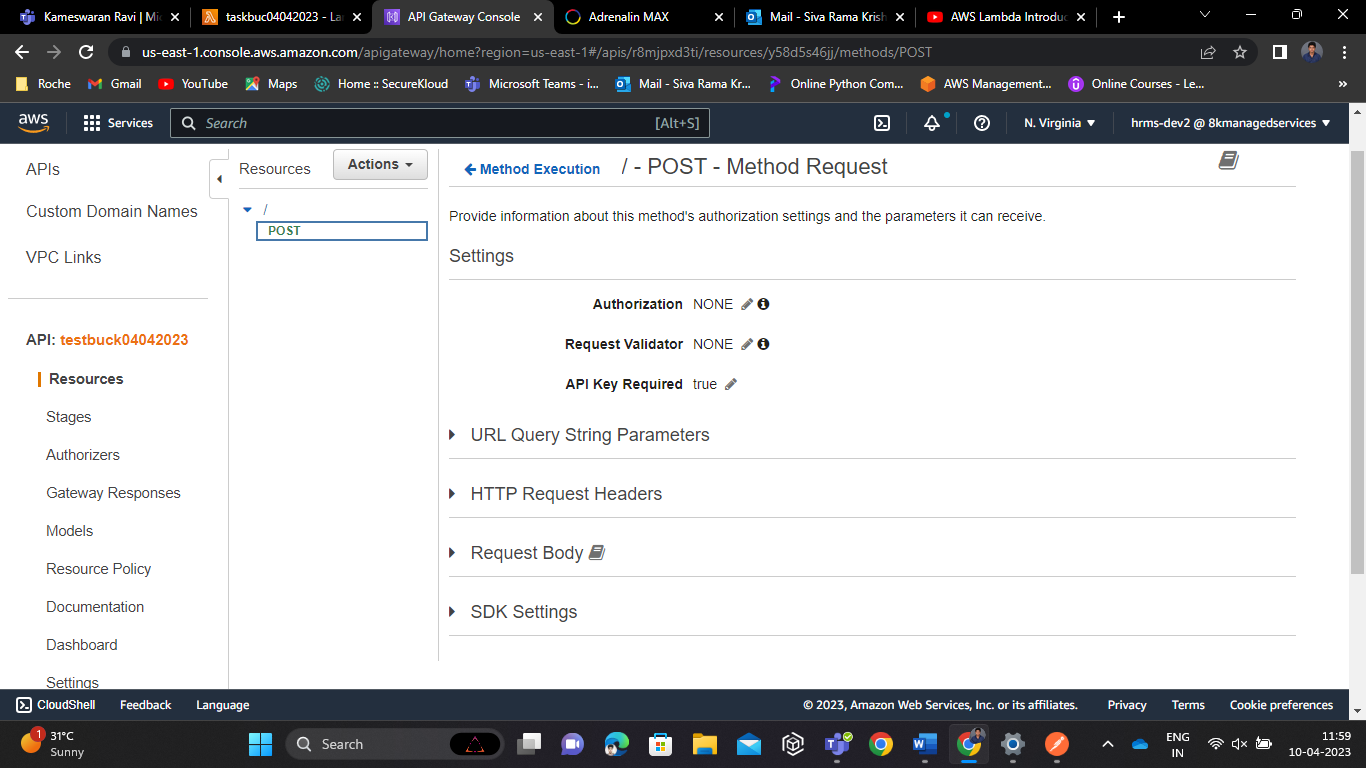
There are so many steps are Available for authentication,

IAM based authentication is one among those.. For this authentication method,

We have to go to the API gateway method page,



And go to the Method request page,



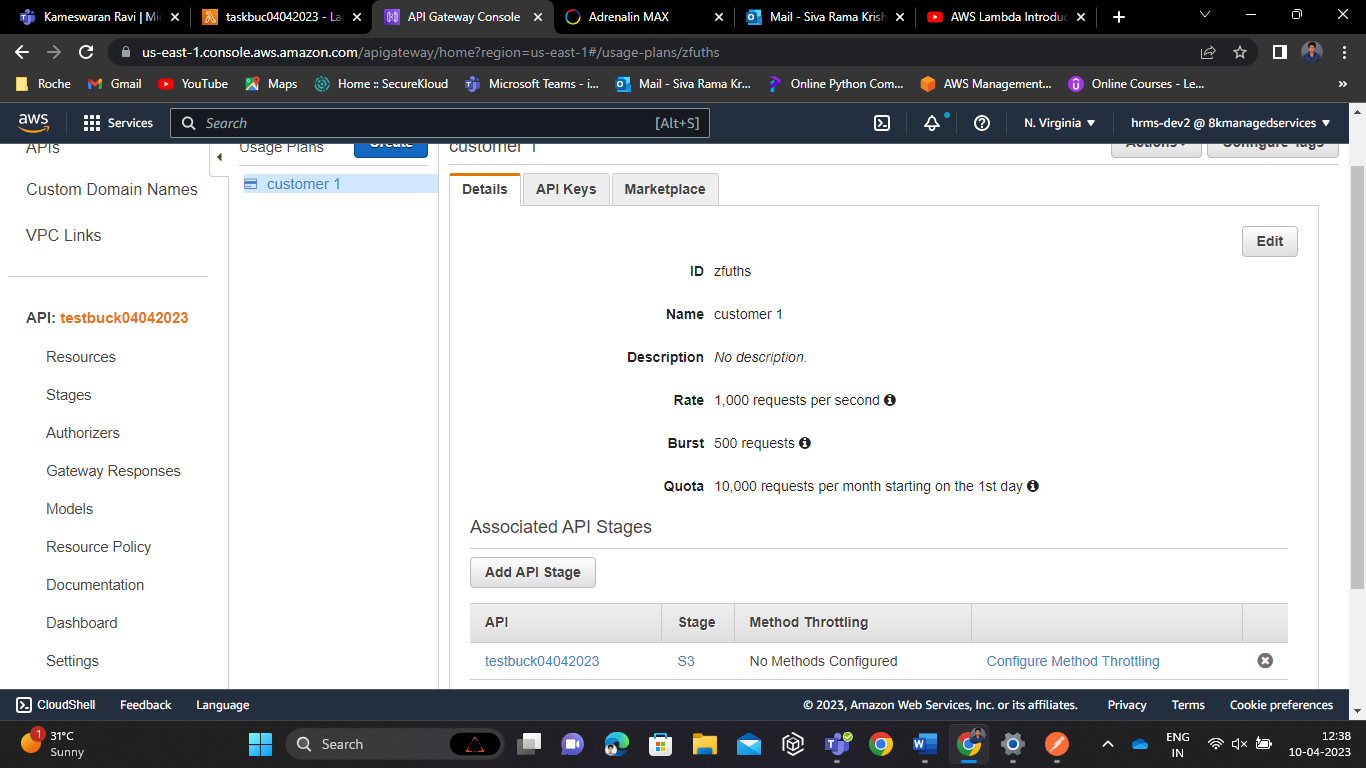
On the API key required parameters we have to put it on as True.

Then create the API KEY for this..

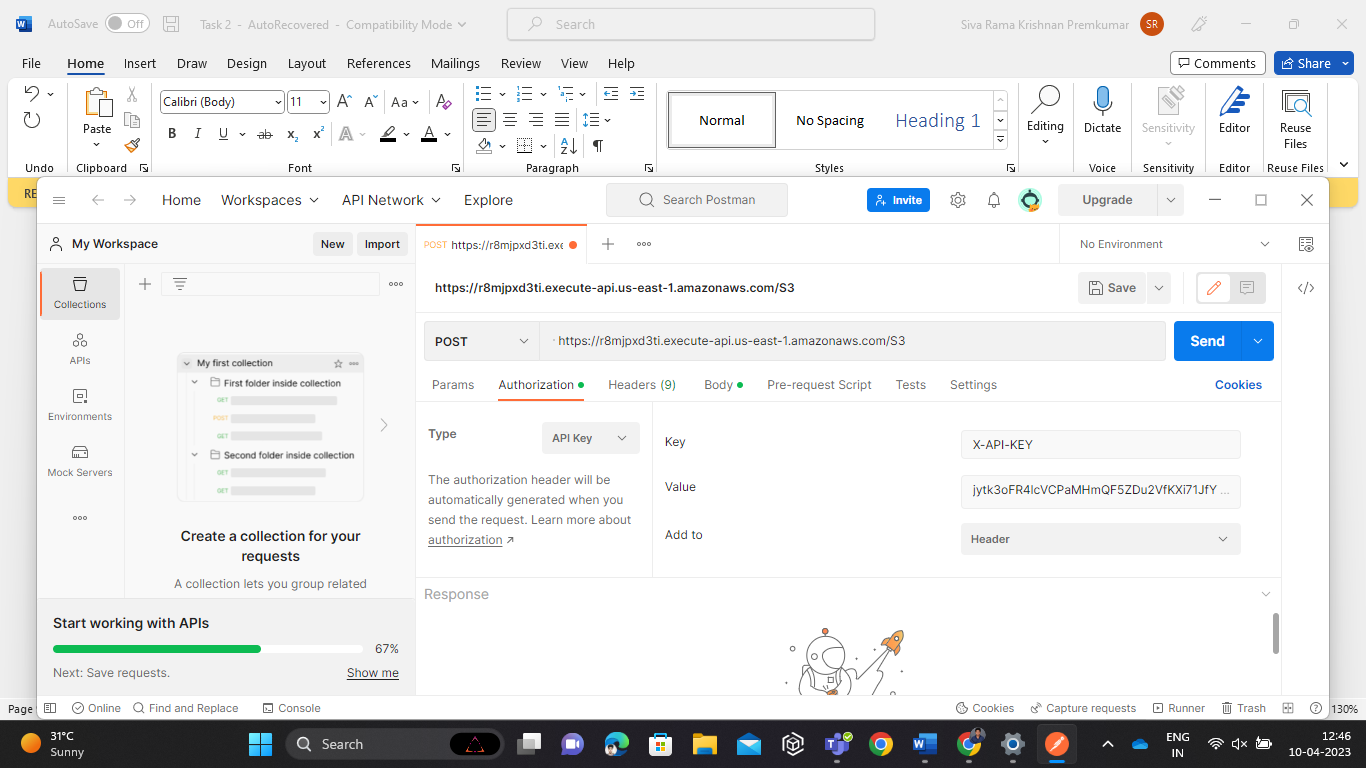
A screenshot of a computer

Description automatically generated

After this create a usage plan, then add the API stage for this usage plan



When we create the API key, by that moment the API invoke URL wont work properly in postman.it shows us our API gateway is authorized under IAM authentication. To overcome this we need use our API key and value in postman application.



By this we can Authenticate our API Gateway ………

------------------------------------------------------------------------------------------------------