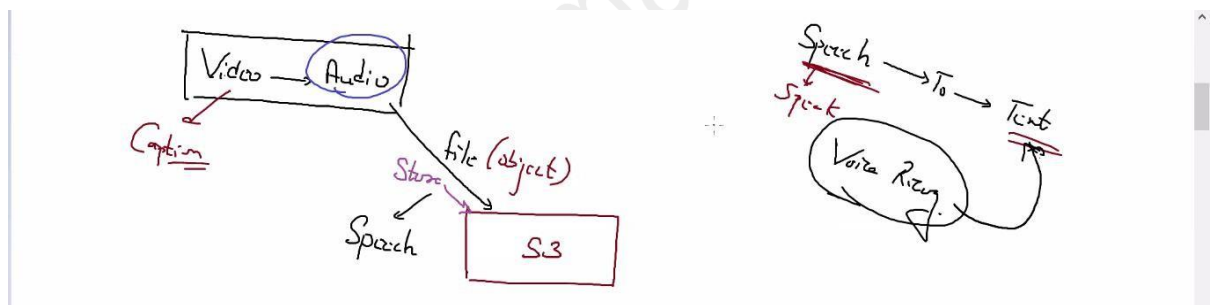




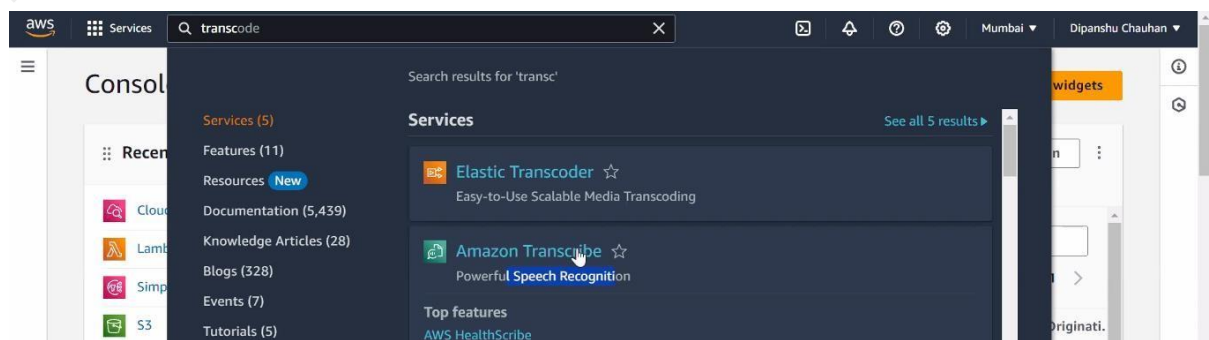
AWS Session 09

Summary 27/03/2024

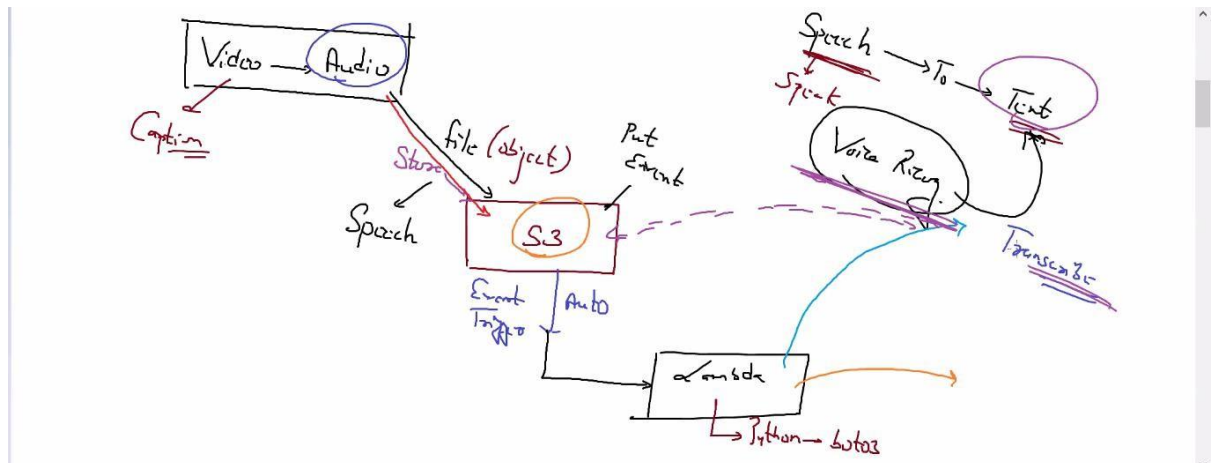
- Today we will create an interesting project in which we will integrate multiple AWS services.
- Let's take an example, Netflix as a company uses the S3 service for the cloud storage, now they have the requirement of as soon as they store any audio or video in the S3 it should be converted into the text.



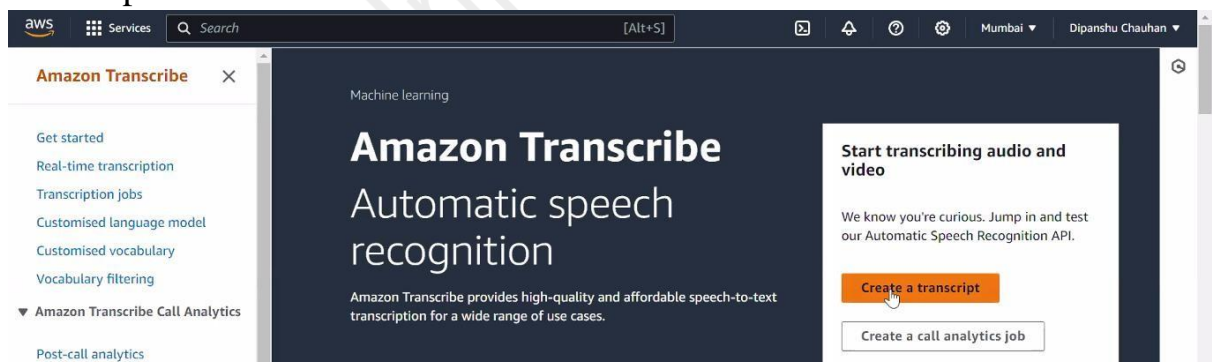
- We want some service who will take the audio and convert it into the text, first it will recognise the voice and then convert it into the text. This is a task of machine learning actually.
- AWS has a service known as the **Amazon Transcribe** which provide the speech to text service.



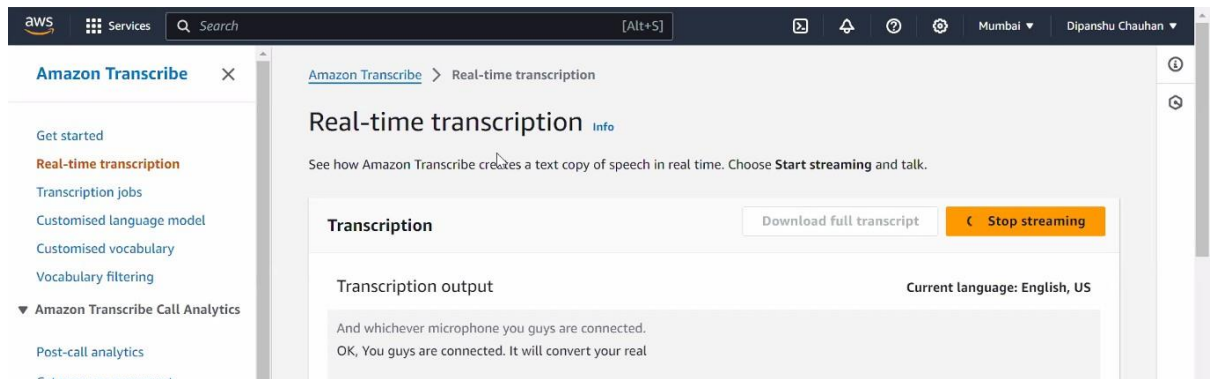
- We will create a S3 trigger by which as soon as any new thing come up in the bucket, it will invoke the lambda function in which we will have a code that will connect to the transcribe service and will convert the speech file into the text.



- Using the Transcribe service.
 - Search for the Transcribe service and click on the create transcript option.



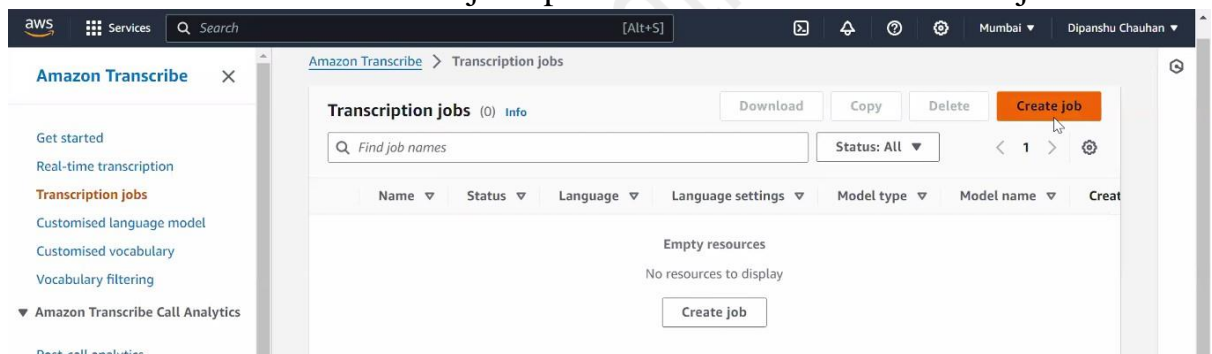
- This service also supports the Real-time transcription in which it will take real time input from the microphone and convert that audio into the text.



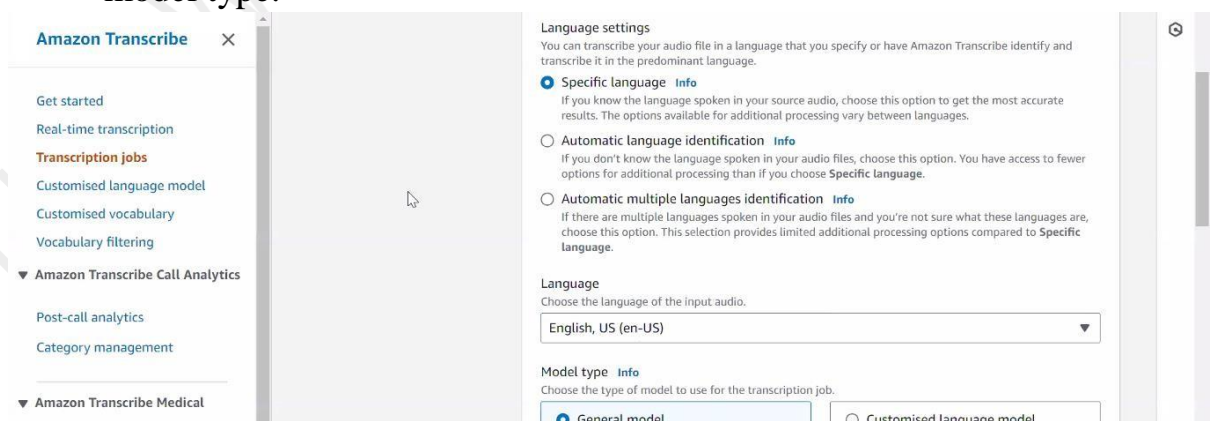
- For our use case we want Transcribe to convert an offline file to the text, for that transcribe have to create a job and to create a job click on the transcription job option.



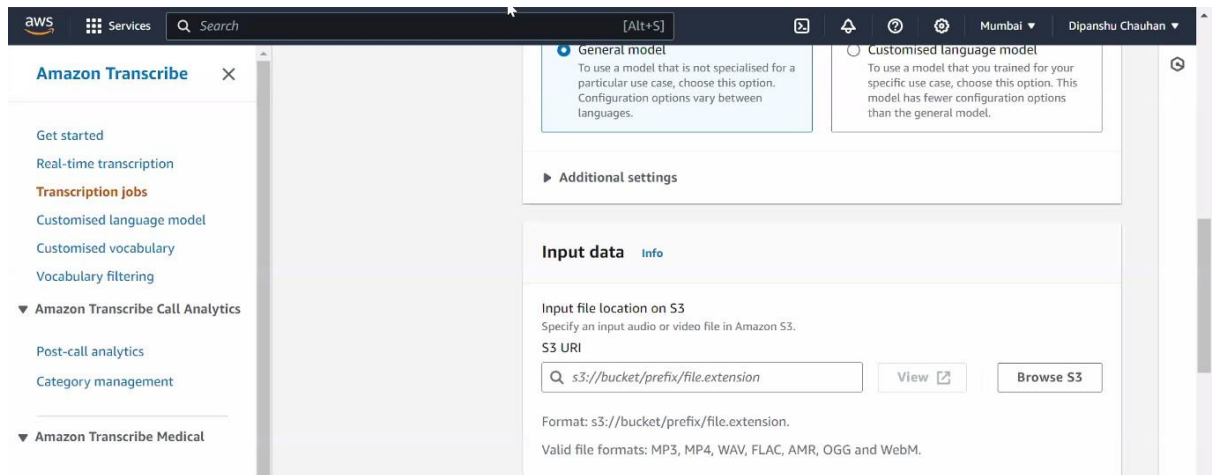
- Then click on the create job option to create a new transcribe job.



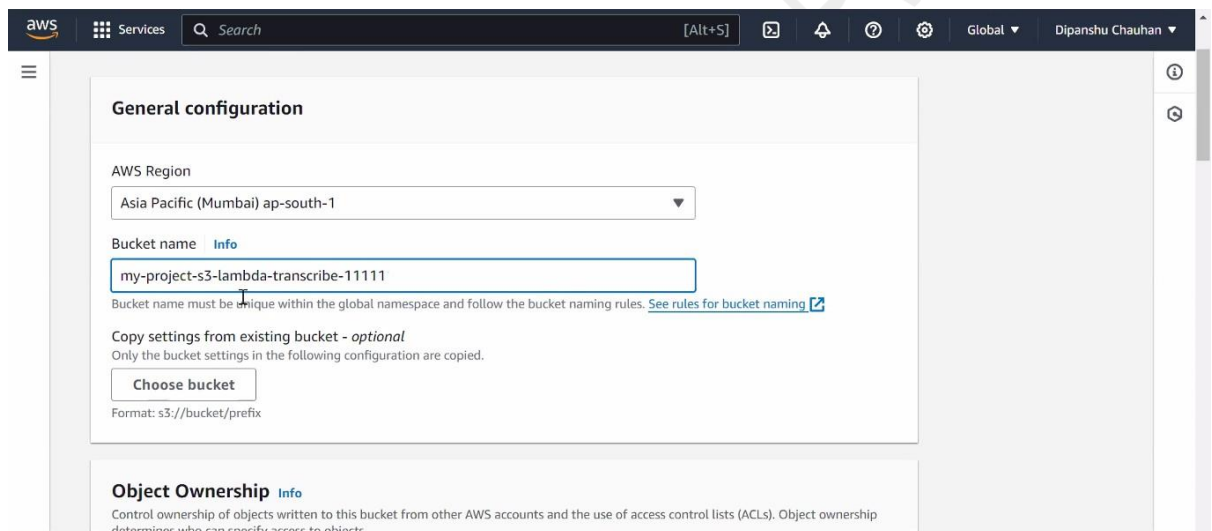
- Give this job a name and also select the language settings and the model type.



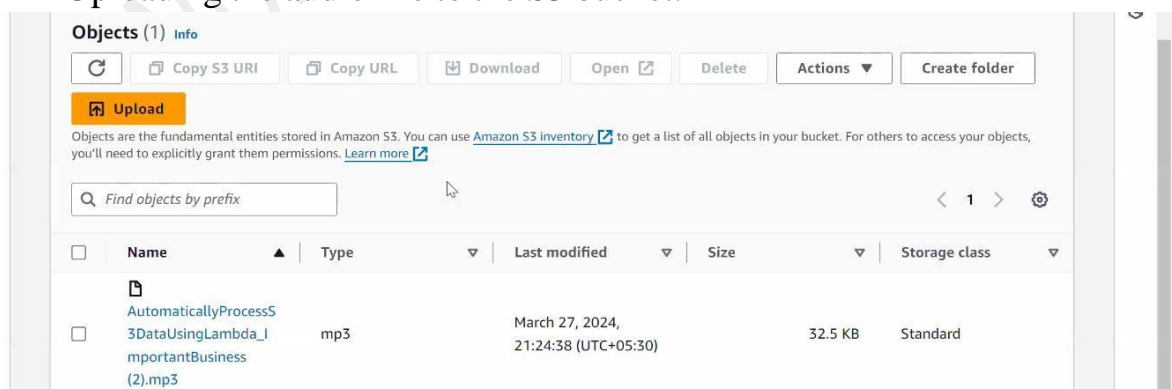
- Transcribe only take the input file from the S3, so first we have to upload the audio file to the S3 bucket.



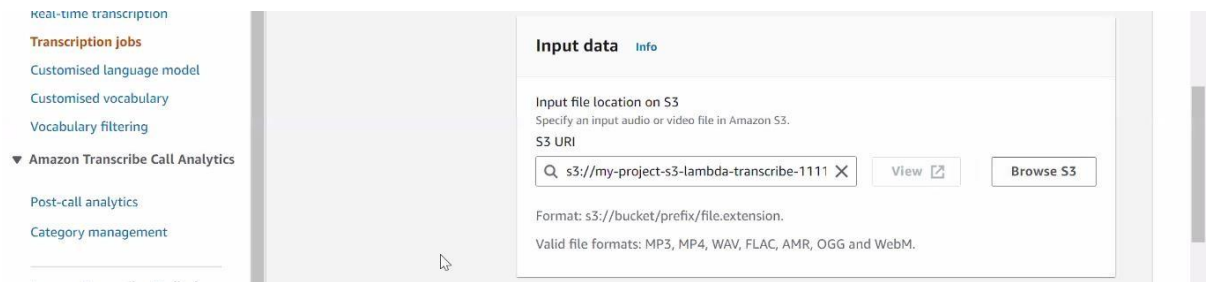
- To upload the data or the object in the S3 we are creating a new bucket, give this bucket a name and this bucket is the one which will have all the audio files.



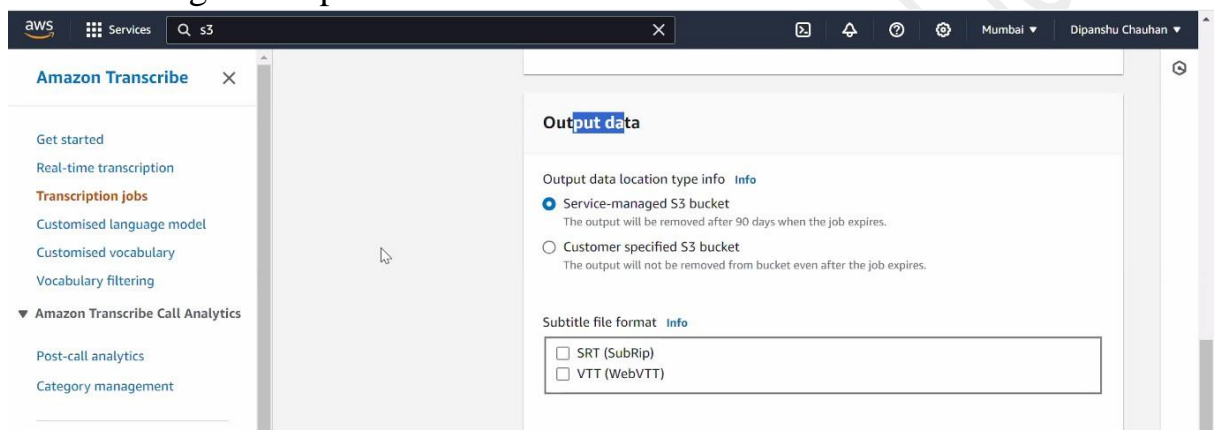
- Uploading the audio file to the S3 bucket.



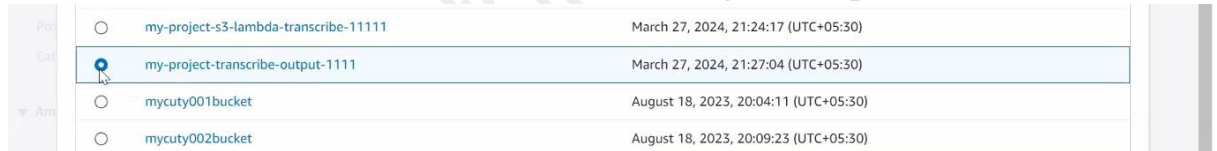
- Now in the Transcribe, we can provide the input file details.



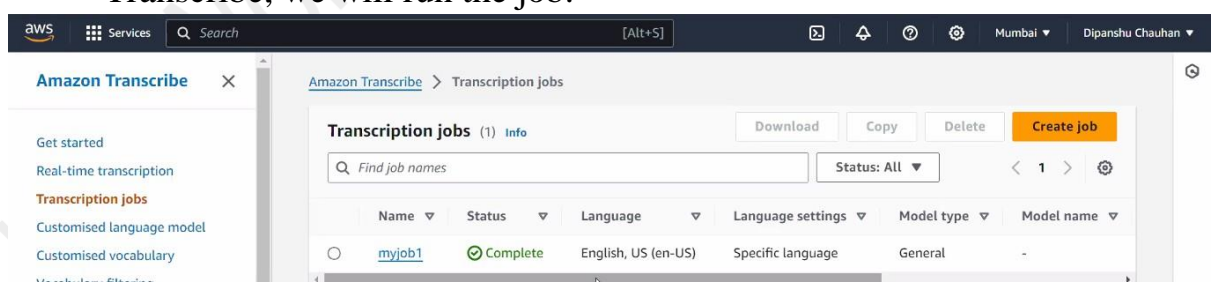
- After converting the Speech file into the text, Transcribe will store it in a file and that file also stores in the S3.
- Either we can use the same bucket that have the audio file for storing the output or we can create a new S3 bucket.



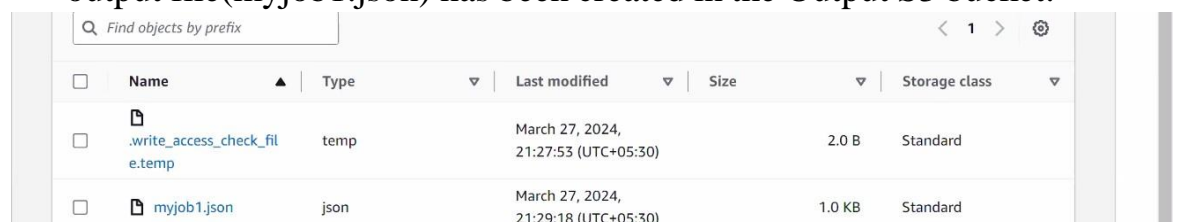
- We will create a new S3 bucket for storing the output files.



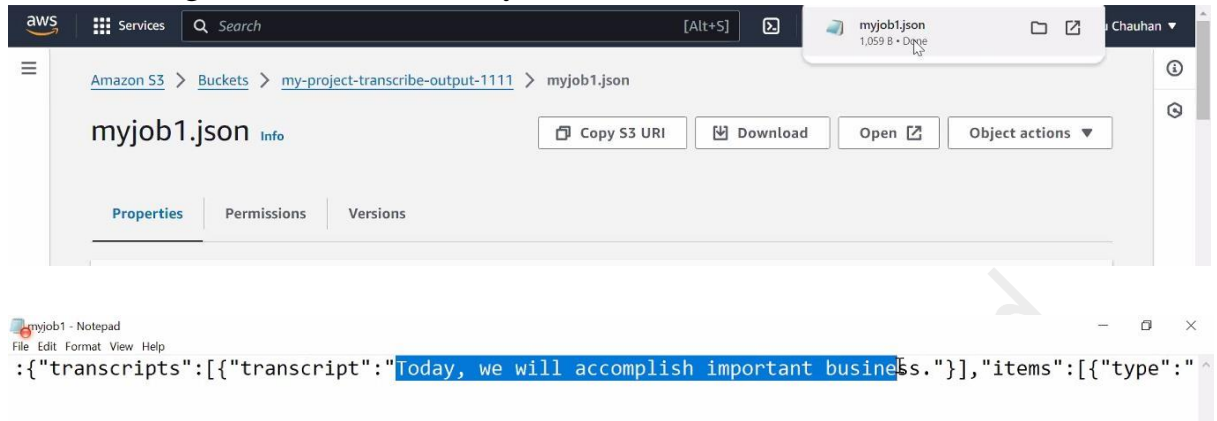
- After providing the bucket details in the Output data of the Transcribe, we will run the job.



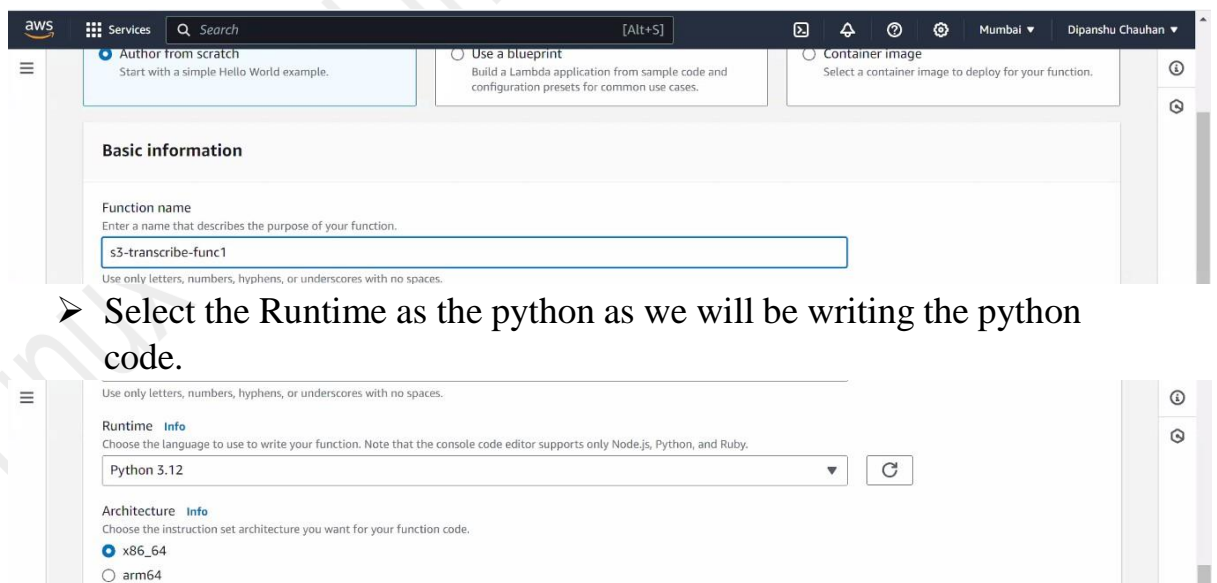
- As soon as the job is successfully completed, we can see a new output file(myjob1.json) has been created in the Output S3 bucket.



- And if we open the output file, we can see that the output text has been generated successfully.

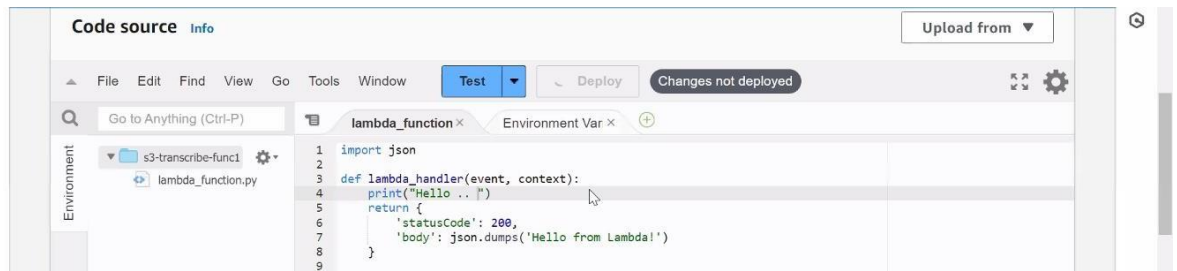


- Now we want this Transcribe job to be run and completed automatically, as soon as the audio is uploaded in the S3 bucket, it will be converted to text automatically.
- We have to use the Event-driven architecture here for this automation.
- Building the architecture or the Project.
 - We will take the bucket first and will integrate it with the lambda.
 - For that we will create a lambda function first.



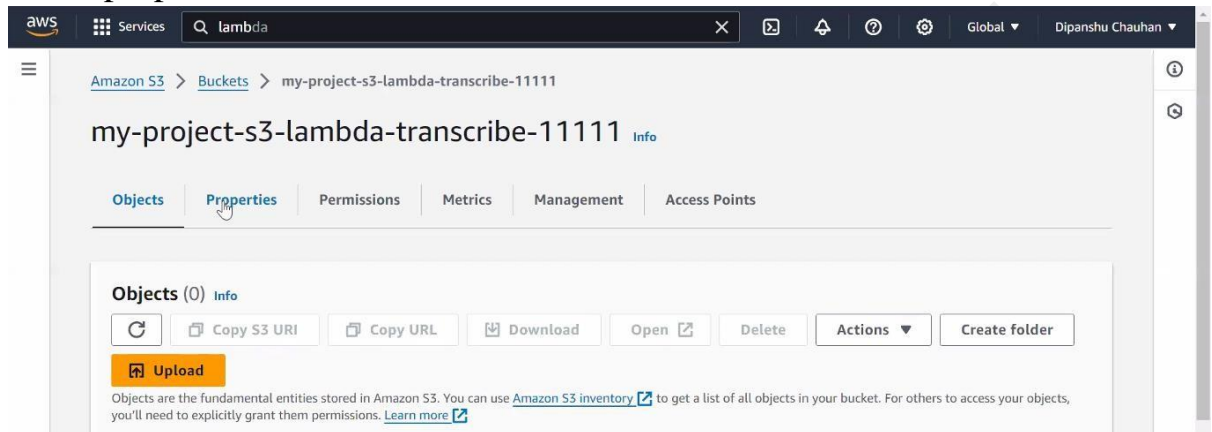
- Select the Runtime as the python as we will be writing the python code.

- Below is the sample code for the testing purpose only.

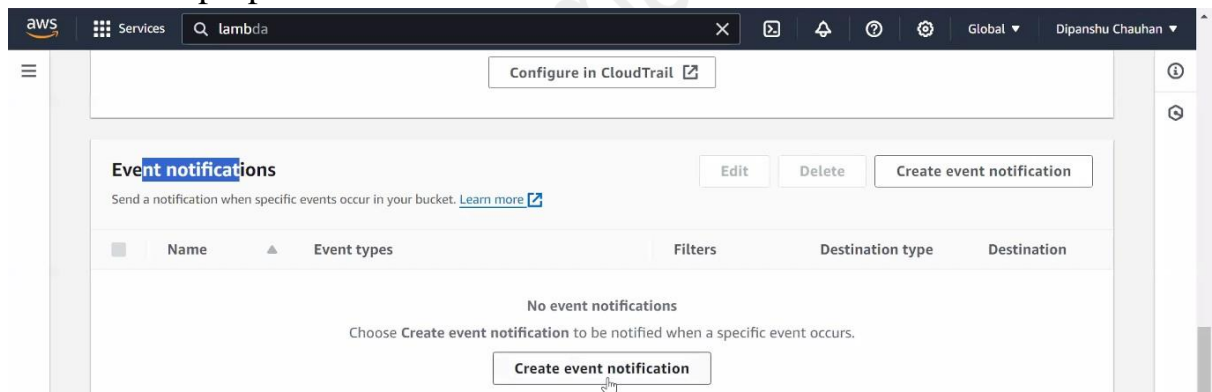


```
1 import json
2
3 def lambda_handler(event, context):
4     print("Hello .. !")
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from Lambda!')}
8
9
```

- Now to integrate the S3 to the Lambda we have to go to the properties of the bucket.



- In the properties we have to set a create event notification.



- Event notification will be, as soon as any **PUT** event occur in the bucket, it will invoke the lambda function.
- Select the Event type as the **PUT** and destination as lambda function which we have created.

Event types
Specify at least one event for which you want to receive notifications. For each group, you can choose an event type for all events, or you can choose one or more individual events.

Object creation

☐ All object create events
s3:ObjectCreated:*

☒ Put
s3:ObjectCreated:Put

☐ Post
s3:ObjectCreated:Post

☐ Copy
s3:ObjectCreated:Copy

☐ Multipart upload completed
s3:ObjectCreated:CompleteMultipartUpload

Destination
Choose a destination to publish the event. [Learn more](#)

☒ **Lambda function**
Run a Lambda function script based on S3 events.

☐ SNS topic
Fanout messages to systems for parallel processing or directly to people.

☐ SQS queue
Send notifications to an SQS queue to be read by a server.

Specify Lambda function

☒ Choose from your Lambda functions

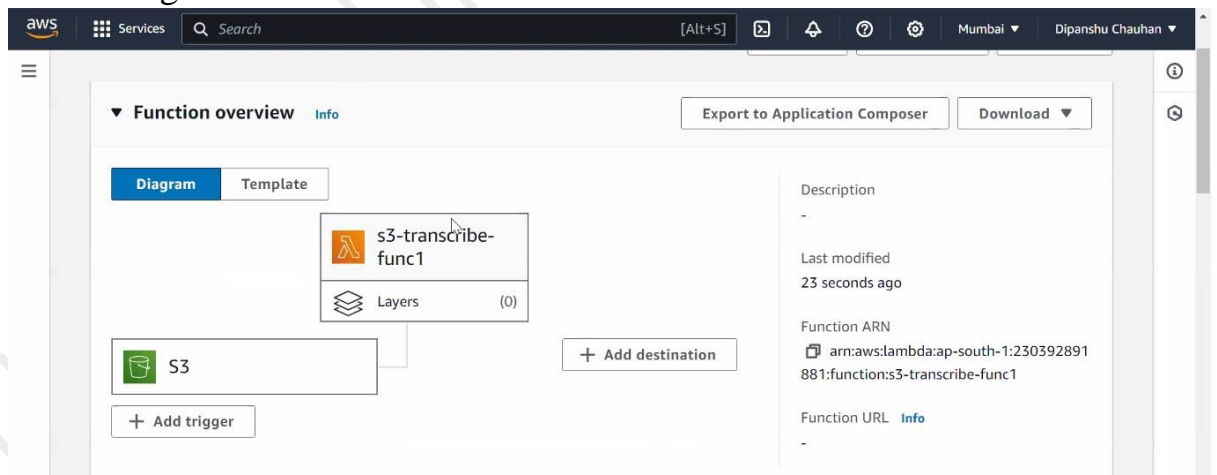
☐ Enter Lambda function ARN

Lambda function

s3-transcribe-func1

Cancel **Save changes**

- Now we can see that the S3 and the lambda is successfully integrated to each other.



- Now we have to integrate the lambda with the transcribe.
- As soon as any new file has been uploaded in the S3 bucket, it will trigger the lambda function and the lambda function will send the file to the transcribe.
- Now lambda should know the name of the audio file before sending it to the Transcribe.

- Whenever any service triggers the lambda, it sends a lot of data or information to the lambda and this data is stored in the **event** variable.

```

1 import json
2
3 def lambda_handler(event, context):
4     print("Hello .. from LW ... ")
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from Lambda!')
8     }
9

```

- In all these information we can see the name of the object that is uploaded in the S3.

```

{
  "Records": [
    {
      "eventVersion": "2.1",
      "eventSource": "aws:s3",
      "awsRegion": "ap-south-1",
      "eventTime": "2024-03-27T16:25:18.534Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "A2G91NGYITEWNJ"
      },
      "requestParameters": {
        "sourceIPAddress": "103.59.75.208"
      },
      "responseElements": {
        "x-amz-request-id": "Z9DS9G7AZ36FWAXC",
        "x-amz-id-2": "MpaIjWe2xgcu616rSHTwEhkQAa1wHtSIMVf4s87/8isUJmbWjF2ytcPV7iLDRDgI7L6tnOzAiD/aHJBnS7YB/Gfgv1PMurfqBZ3psBSk6g="
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "myrule1",
        "bucket": {
          "name": "my-project-s3-lambda-transcribe-11111",
          "ownerIdentity": {
            "principalId": "A2G91NGYITEWNJ",
            "arn": "arn:aws:s3:::my-project-s3-lambda-transcribe-11111"
          },
          "object": {
            "key": "Screen+Amit+Shah.png",
            "size": 1594131,
            "eTag": "b2af0087babb8e2336d0cc0d4bde4683",
            "sequencer": "006604486E6CDE5CA2"
          }
        }
      }
    }
  ]
}

```

- Now we have to somehow retrieve or extract this name from the information.
- For this we will use the python dictionary logic.
- Let's say we have stored all the information in a db variable and we will apply some operation on it.
- In the Records we can see that the S3 is present.

```

In [1]: db = {'Records': [{'eventVersion': '2.1', 'eventSource': 'aws:s3', 'awsRegion': 'ap-south-1',
2
Out[3]: [{'eventVersion': '2.1',
          'eventSource': 'aws:s3',
          'awsRegion': 'ap-south-1',
          'eventTime': '2024-03-27T16:25:18.534Z',
          'eventName': 'ObjectCreated:Put',
          'userIdentity': {'principalId': 'A2G91NGYITEWNJ'},
          'requestParameters': {'sourceIPAddress': '103.59.75.208'},
          'responseElements': {'x-amz-request-id': 'Z9DS9G7AZ36FWAXC',
                                'x-amz-id-2': 'MpaIjWe2xgcu616rSHTwEhkQAa1wHtSIMVf4s87/8isUJmbWjF2ytcPV7iLDRDgI7L6tnOzAiD/aHJBnS7YB/Gfgv1PMurfqBZ3psBSk6g='},
          's3': {'s3SchemaVersion': '1.0',
                  'configurationId': 'myrule1',
                  'bucket': {'name': 'my-project-s3-lambda-transcribe-11111',

```

- Here we have extracted the S3 details and now from the S3 we have to extract the bucket name.

```
In [5]: 1 db['Records'][0]['awsRegion']
Out[5]: 'ap-south-1'

In [7]: 1 db['Records'][0]['s3']
Out[7]: {'s3SchemaVersion': '1.0',
'configurationId': 'myrule1',
'bucket': {'name': 'my-project-s3-lambda-transcribe-11111',
'ownerIdentity': {'principalId': 'A2G91NGVITEWNJ'},
'arn': 'arn:aws:s3:::my-project-s3-lambda-transcribe-11111'},
'object': {'key': 'Screen+Amit+Shah.png',
'size': 1594131,
'eTag': 'b2af0087babb8e2336d0cc0d4bde4683',
'sequenceNumber': '006604486E6CDE5CA2'}}
```

```
In [ ]: 1
```

```
In [5]: 1 db['Records'][0]['awsRegion']
Out[5]: 'ap-south-1'

In [9]: 1 bucketName= db['Records'][0]['s3']['bucket']['name']
Out[9]: 'my-project-s3-lambda-transcribe-11111'
```

- In the S3, file name is known as the **object key** so for the file name we have to retrieve the object key.

```
In [16]: 1 bucketName= db['Records'][0]['s3']['bucket']['name']

In [15]: 1 fileName = db['Records'][0]['s3']['object']['key']

In [17]: 1 bucketName
Out[17]: 'my-project-s3-lambda-transcribe-11111'

In [18]: 1 fileName
Out[18]: 'Screen+Amit+Shah.png'
```

- In the Transcribe service we have to provide the S3 URI also and we can create this using the python.
- Below is the format of the S3 URI, we will create this using the python.

Get started

Real-time transcription

Transcription jobs

Customised language model

Customised vocabulary

Vocabulary filtering

Amazon Transcribe Call Analytics

Post-call analytics

Category management

Input data Info

Input file location on S3
Specify an input audio or video file in Amazon S3.

S3 URI
s3://bucket/prefix/file.extension View Browse S3

Format: s3://bucket/prefix/file.extension.
Valid file formats: MP3, MP4, WAV, FLAC, AMR, OGG and WebM.

- Now finally we have fulfilled all the requirements and those are bucket name, file name and the S3 URI.

Jupyter Notebook interface showing code execution. The notebook is titled 'Untitled1' and shows the following code cells:

```

In [17]: 1 bucketName
Out[17]: 'my-project-s3-lambda-transcribe-11111'

In [18]: 1 fileName
Out[18]: 'Screen+Amit+Shah.png'

In [20]: 1 finalURL = "s3://" + bucketName + "/" + fileName

In [ ]: 1 finalURL
        2

```

➤ Now we have to run this code in the lambda.

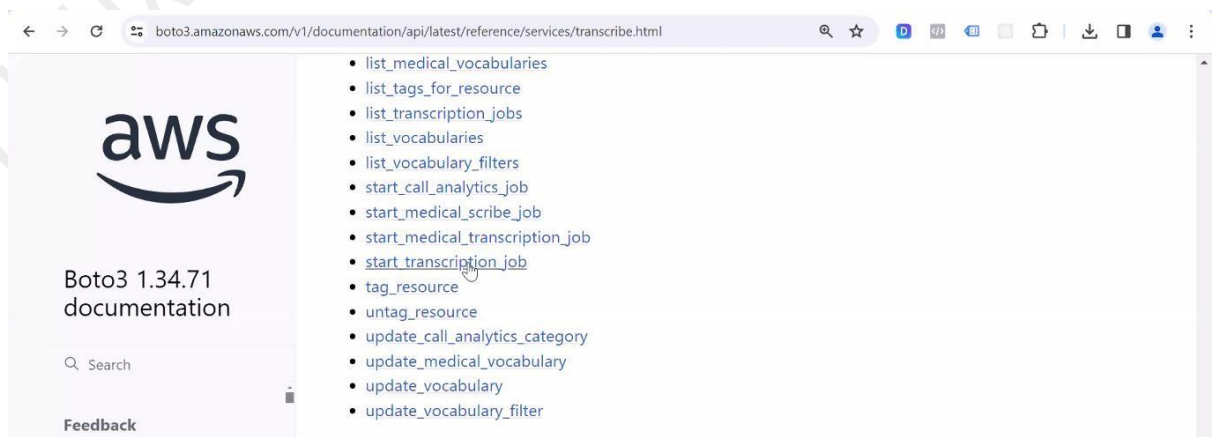
AWS Lambda console showing the function `s3-transcribe-func1`. The code is as follows:

```

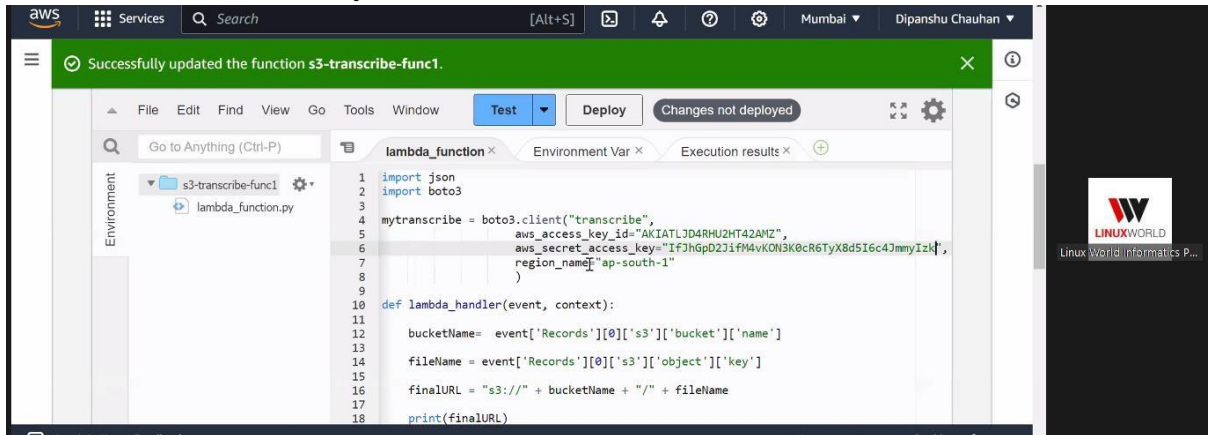
1 import json
2
3 def lambda_handler(event, context):
4
5     bucketName= event['Records'][0]['s3']['bucket']['name']
6     fileName = event['Records'][0]['s3']['object']['key']
7
8     finalURL = "s3://" + bucketName + "/" + fileName
9
10    print(finalURL)
11
12
13    print("Hello .. from LW ... ")
14    print(event)
15    return {
16        'statusCode': 200,
17        'body': json.dumps('Hello from Lambda!')}
18
19

```

- Now lambda know the file name so it can contact to the transcribe easily and provide the audio file to it.
- Now we have to write the code for connecting the lambda to the transcribe.
- We will use the boto3 code to connect the lambda with the transcribe.
- We can see the boto3 code for creating the transcription job in the documentation also.



- Import the boto3 library in the code and provide the Access key and the secret key.

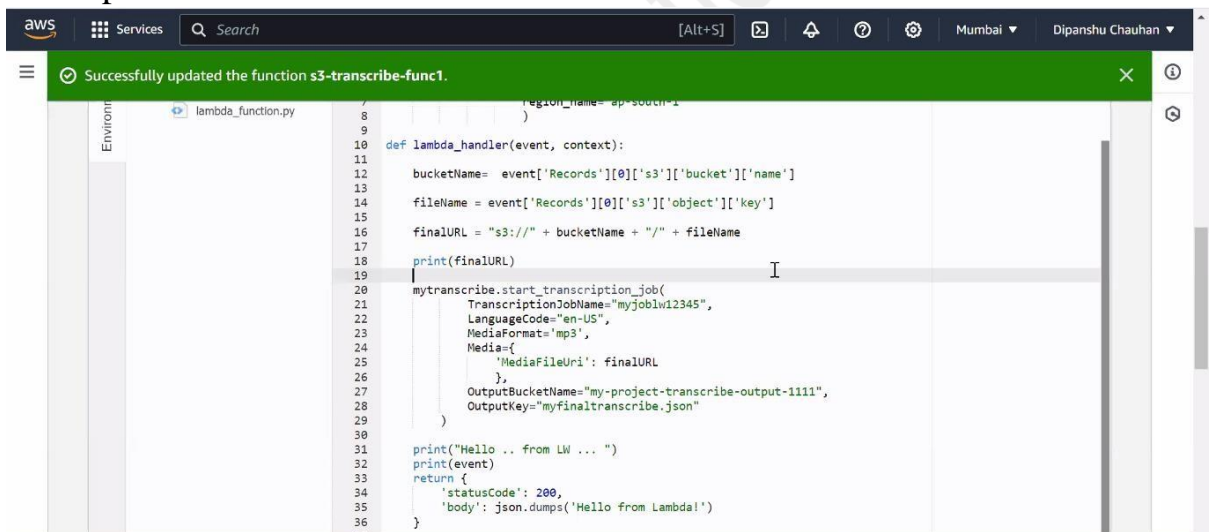


```

1 import json
2 import boto3
3
4 mytranscribe = boto3.client("transcribe",
5                             aws_access_key_id="AKIAIATL3D4RHU2HT42AMZ",
6                             aws_secret_access_key="If3hGpD23ifM4vKON3K0cR6TyX8d5I6c43mmyIzk",
7                             region_name="ap-south-1")
8
9
10 def lambda_handler(event, context):
11
12     bucketName= event['Records'][0]['s3']['bucket']['name']
13
14     fileName = event['Records'][0]['s3']['object']['key']
15
16     finalURL = "s3://" + bucketName + "/" + fileName
17
18     print(finalURL)

```

- For the transcribe job we will need some parameters like the transcription job name, language code, media format, output bucket name and output file name.
- Below is the code in which we have passed all the required parameters.

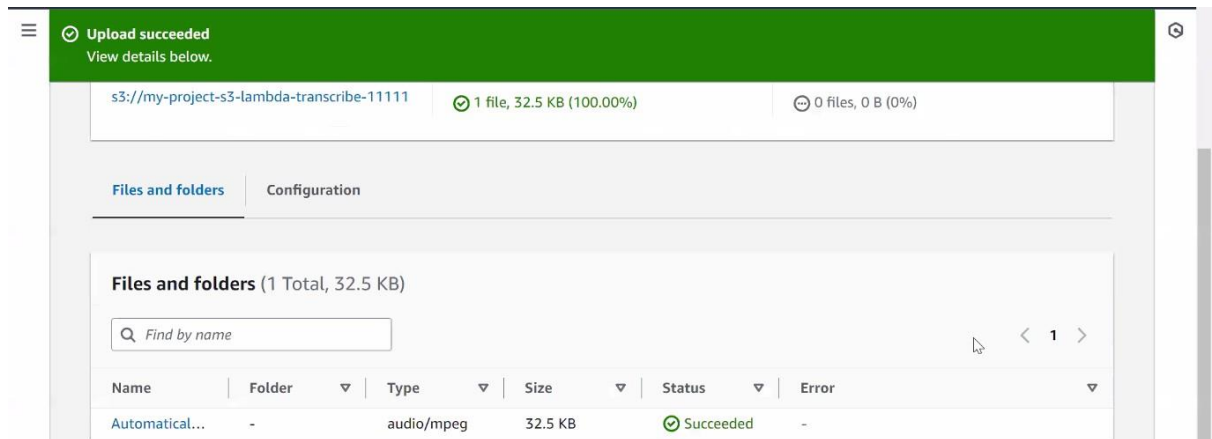


```

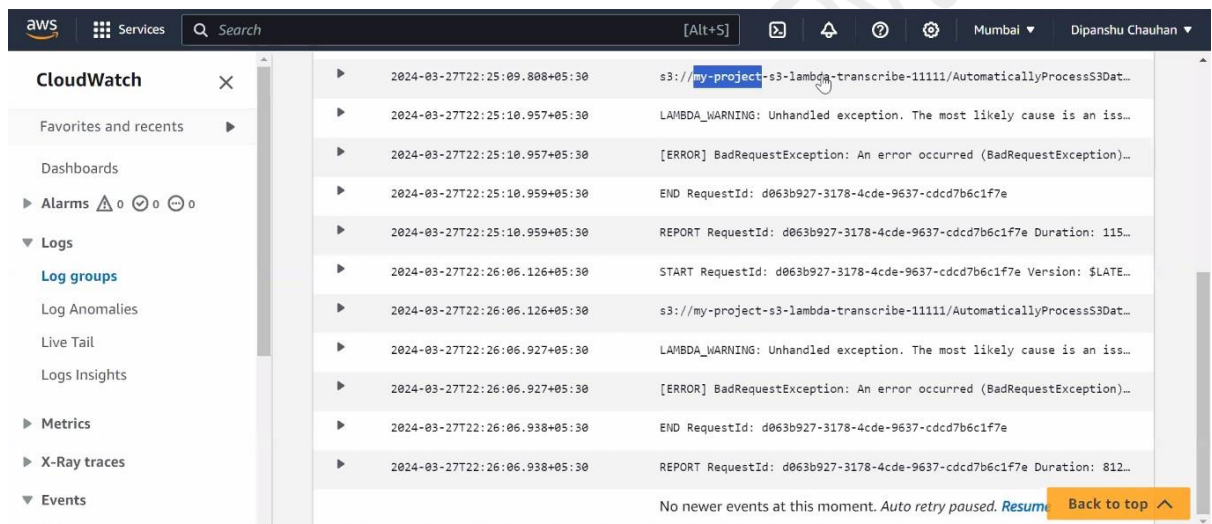
7         region_name="ap-south-1")
8
9
10 def lambda_handler(event, context):
11
12     bucketName= event['Records'][0]['s3']['bucket']['name']
13
14     fileName = event['Records'][0]['s3']['object']['key']
15
16     finalURL = "s3://" + bucketName + "/" + fileName
17
18     print(finalURL)
19
20     mytranscribe.start_transcription_job(
21         TranscriptionJobName="myJob1w12345",
22         LanguageCode="en-US",
23         MediaFormat='mp3',
24         Media={
25             'MediaFileUri': finalURL
26         },
27         OutputBucketName="my-project-transcribe-output-1111",
28         OutputKey="myfinaltranscribe.json"
29     )
30
31     print("Hello .. from LW ... ")
32     print(event)
33     return {
34         'statusCode': 200,
35         'body': json.dumps('Hello from Lambda!')}
36
37

```

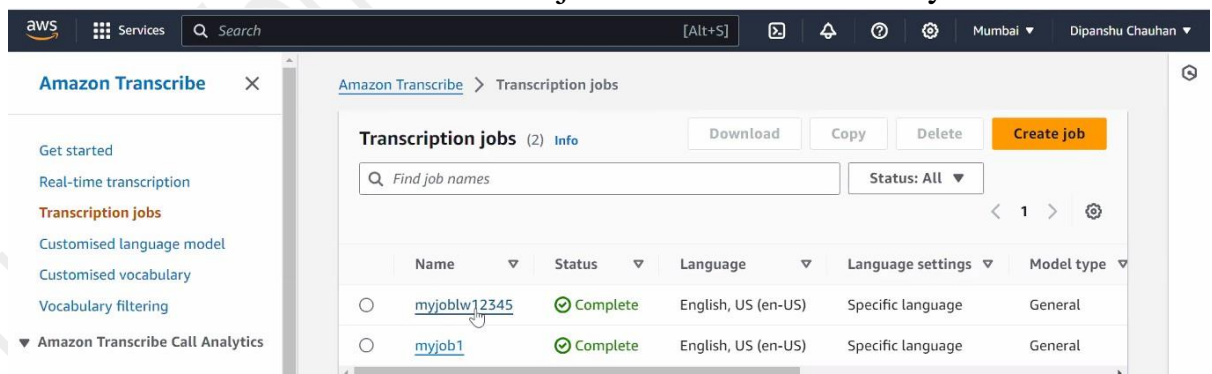
- Now we have successfully written the code that will help the lambda to connect with the transcribe service.
- All we have to do is upload a new file into the S3 bucket and as soon as the file will be uploaded, the lambda function will be invoked and inside the lambda function we have the boto3 code written that will help to create a new transcribe job and the audio file will be converted to the text automatically.



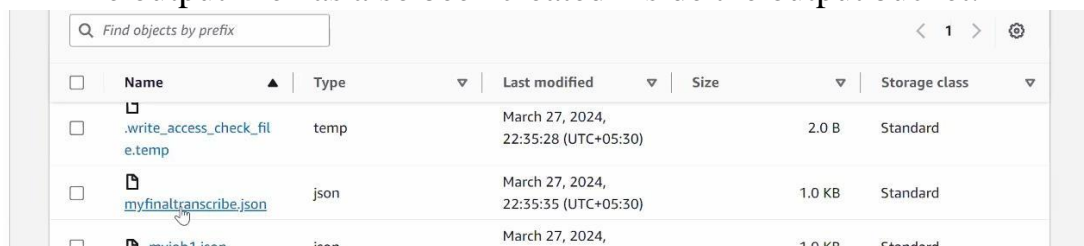
- There might be a possibility that this error come up. To solve this we just have to make sure that our file doesn't have any special character in its name.



- Otherwise we can see that our job is created successfully.



- The output file has also been created inside the output bucket.



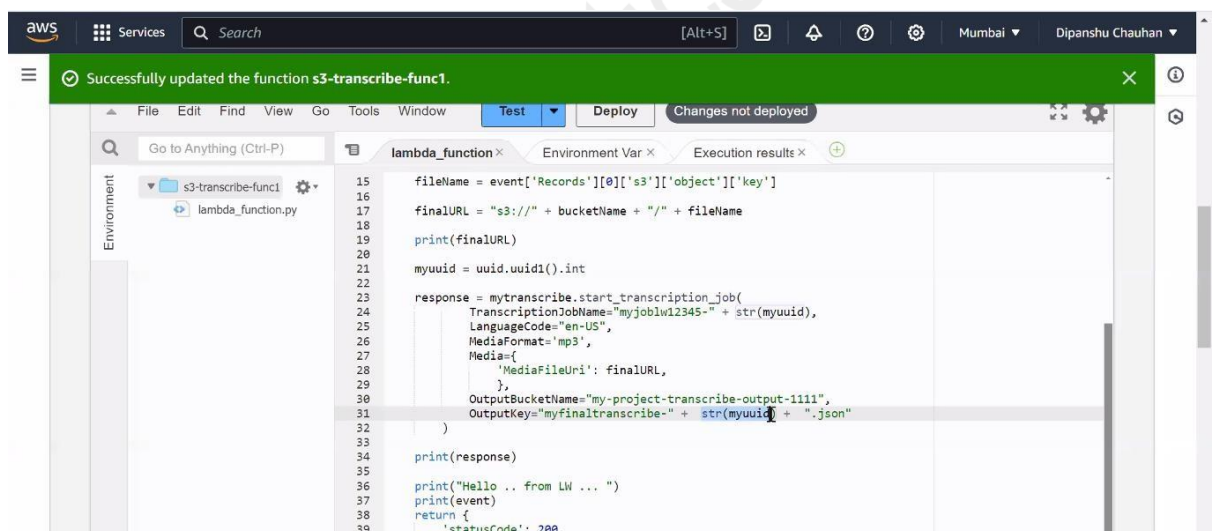
- If we again try to run the job by uploading a new file, it will fail because we have hard-coded the job name in the code and this job already exists.
- So, to overcome this problem we need to have the unique job name for every time we upload a new file.
- For this we can use the **uuid** module of the python, this module helps us in providing the unique ids.

```
In [22]: 1 import uuid

In [24]: 1 uuid.uuid1()

Out[24]: UUID('bde826e2-ec5c-11ee-98e2-b46bfcdc4d66')
```

- Using the same approach in our boto3 code.
- Now every time we upload a new file in the bucket, a new job will be created with a unique id and the output file will also have the unique id.



```
aws
Services
Search
[Alt+S]
Mumbai
Dipanshu Chauhan

Successfully updated the function s3-transcribe-func1.

File Edit Find View Go Tools Window Test Deploy Changes not deployed

Go to Anything (Ctrl-P)

Environment
s3-transcribe-func1
lambda_function.py

15 fileName = event['Records'][0]['s3']['object']['key']
16
17 finalURL = "s3://" + bucketName + "/" + fileName
18
19 print(finalURL)
20
21 myuuid = uuid.uuid1().int
22
23 response = mytranscribe.start_transcription_job(
24     TranscriptionJobName="myjoblw12345-" + str(myuuid),
25     LanguageCode="en-US",
26     MediaFormat='mp3',
27     Media={
28         'MediaFileUri': finalURL,
29     },
30     OutputBucketName="my-project-transcribe-output-1111",
31     OutputKey="myfinaltranscribe-" + str(myuuid) + ".json"
32 )
33
34 print(response)
35
36 print("Hello .. from LW ... ")
37 print(event)
38 return {
39     'statusCode': 200,
```