



AIでコンタクトセンターの 通話録音解析ハンズオンセミナー

2020/11/4

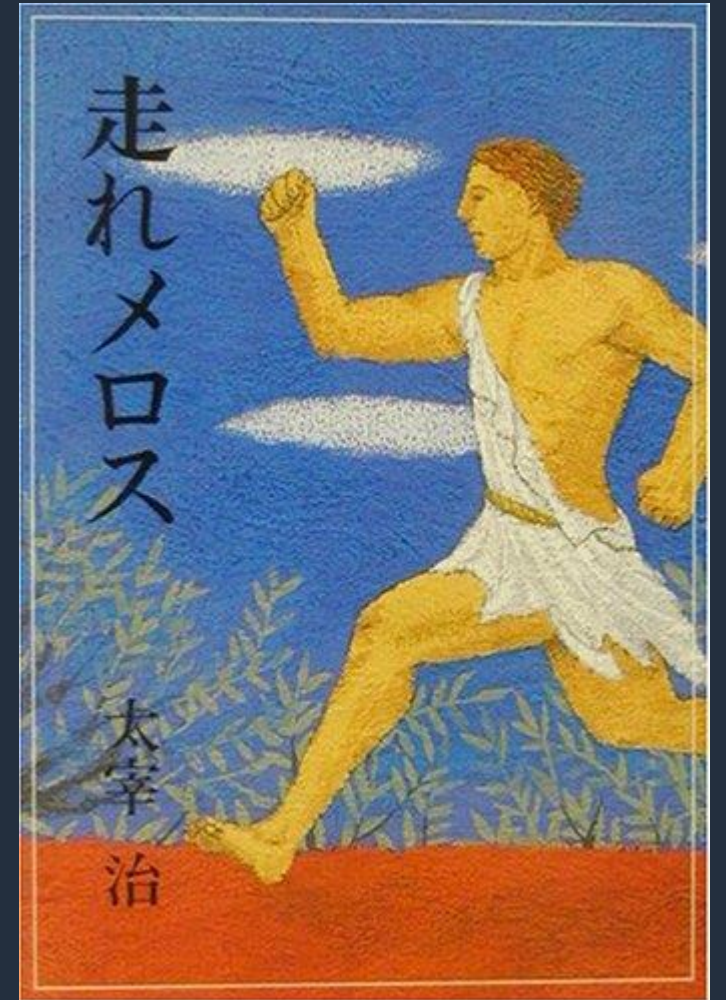
呉 和仁（機械学習ソリューションアーキテクト）

自己紹介

呉 和仁

機械学習ソリューションアーキテクト

- テキスト解析 AI サービス Amazon Comprehend で本を読まずに読書感想文に挑戦してみる
- 本ハンズオンは上記ブログから着想を得たソリューションをデプロイします



本日のコンテンツとゴール

コンテンツ

1. Amazon Transcribe, Amazon Comprehend を AWS SDK for Python boto3 を利用して実行する
2. Voice of Customer Integration Solution for handson をデプロイして、音声ファイルの分析を自動で行い可視化する

ゴール

1. お客様が抱える音声データを、お客様の環境で簡単に分析できるようデプロイ方法を学ぶ
2. 必要に応じてソリューションをカスタマイズできるよう、ソリューションの概要を理解する

ハンズオン手順書

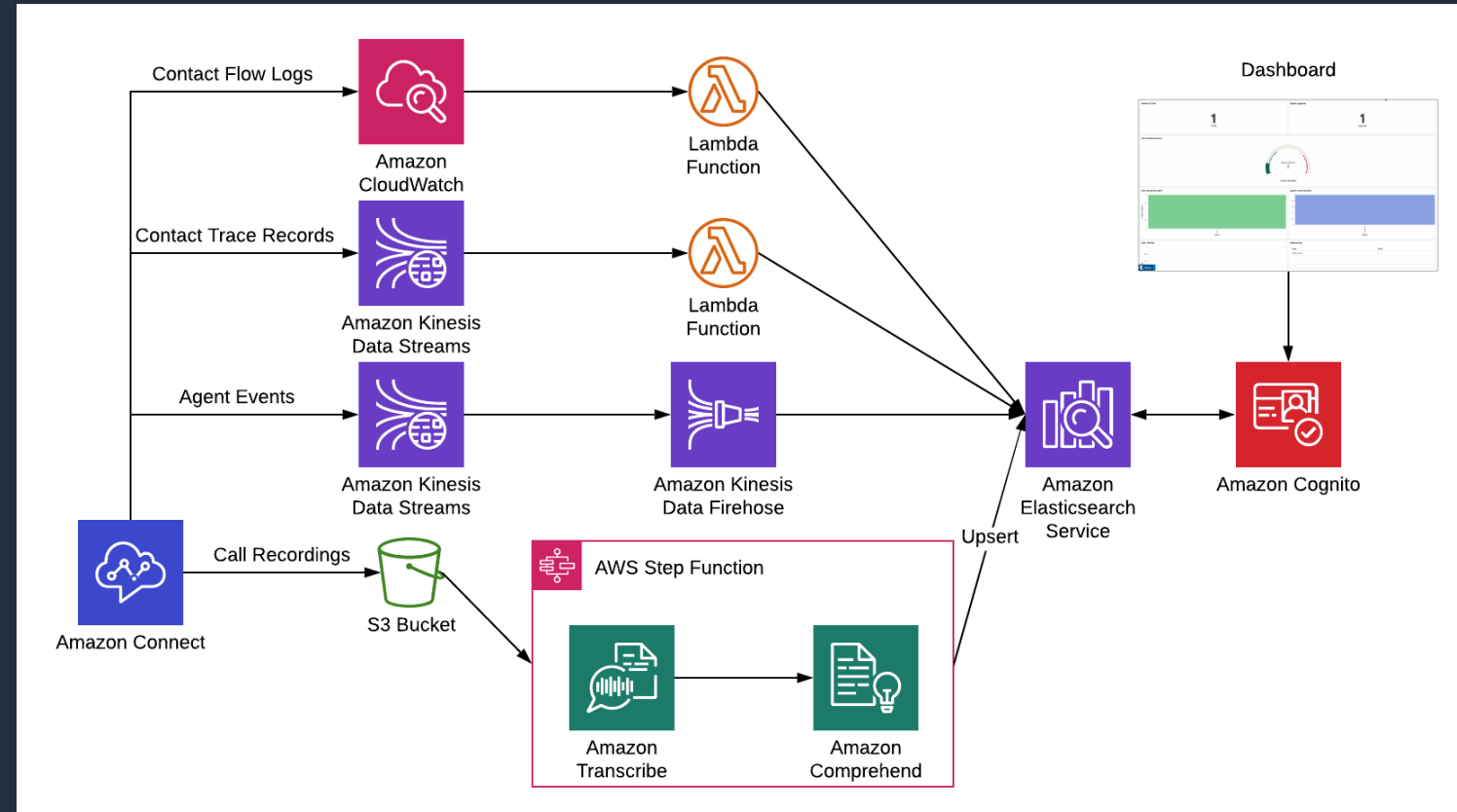
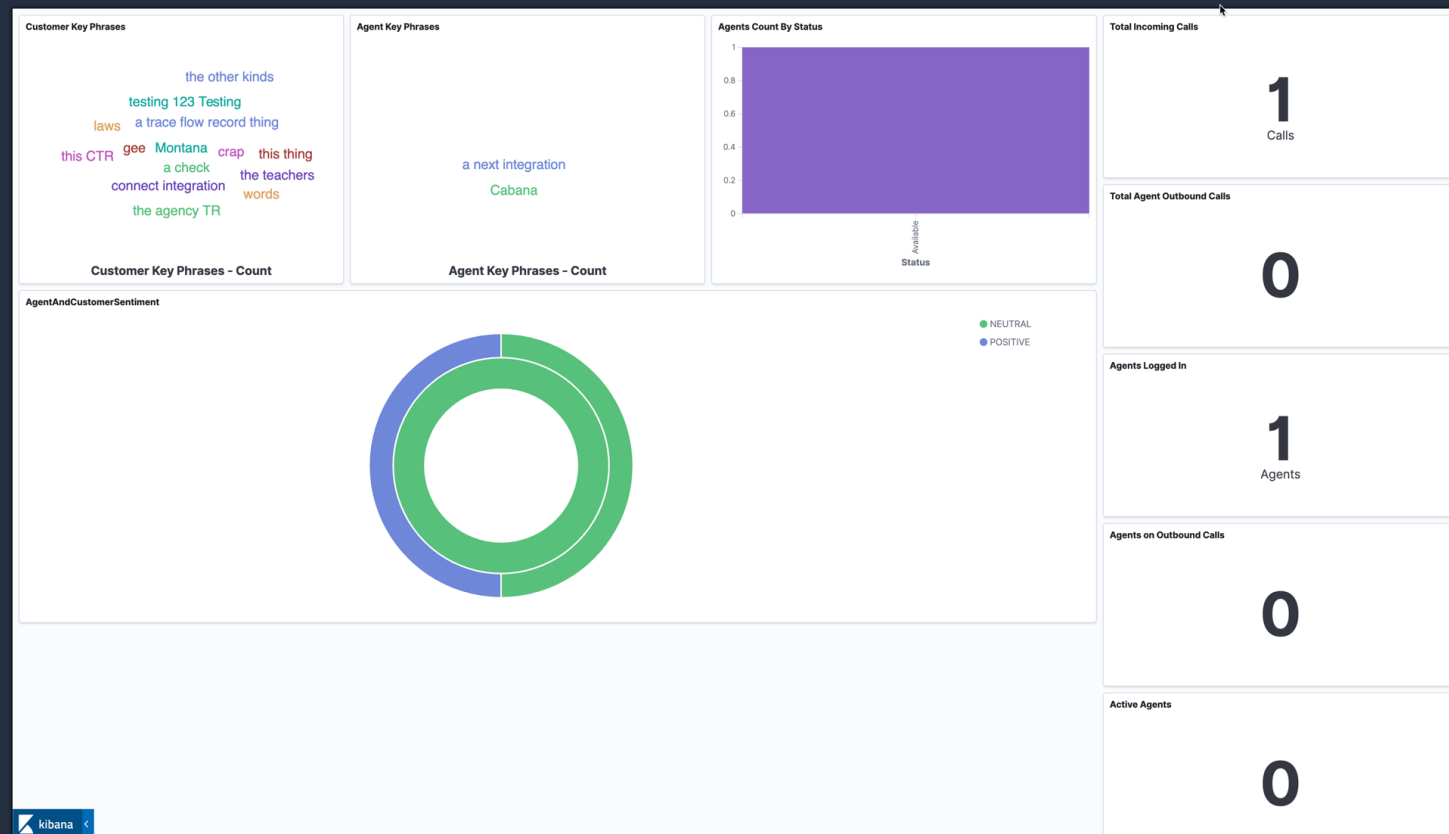
下記リンクにアクセスをお願いします

<https://github.com/kazuhitogo/voice-of-customer-integration-handson>

Star / Fork 歓迎です

Voice of the customer integrations(オリジナル)

- Github [amazon-connect/voice-of-the-customer-integrations](https://github.com/amazon-connect/voice-of-the-customer-integrations)
- Connectの音声データをTranscribeとComprehendで分析、問い合わせ追跡レコードなどのデータもAmazon Elasticsearchに取り込み Kibana で可視化

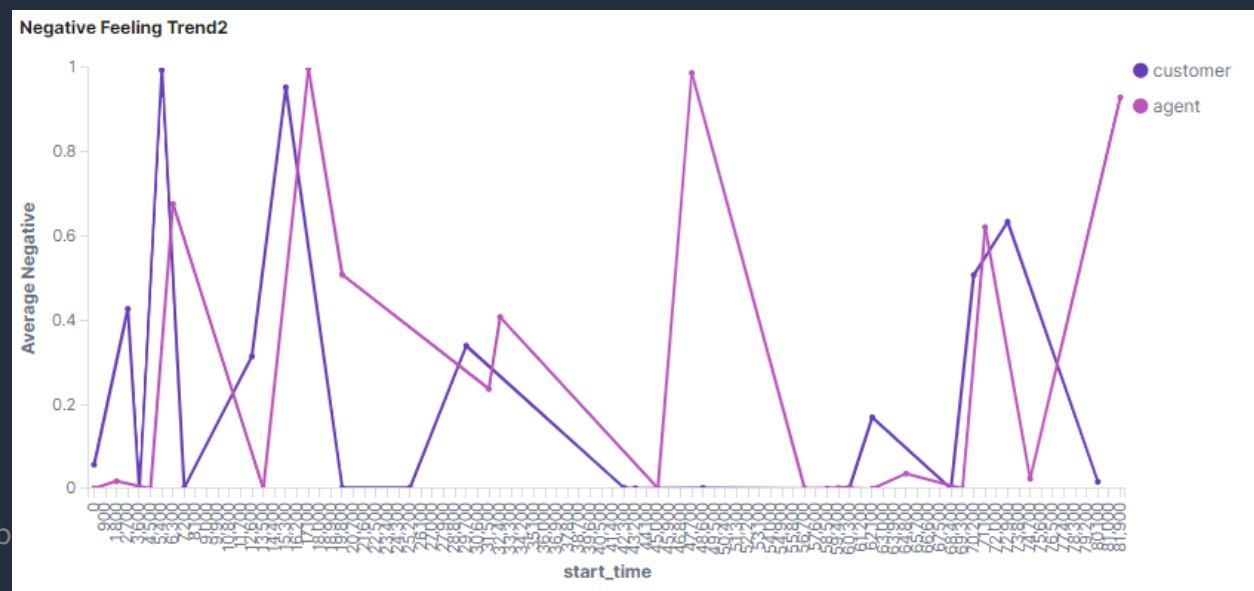


Voice of the customer integrations (カスタマイズ)

- <https://github.com/kazuhitogo/voice-of-the-customer-integrations>
- お客様から頂いた要望に対していくつか機能追加
 - スタッフの名前の取得と表示

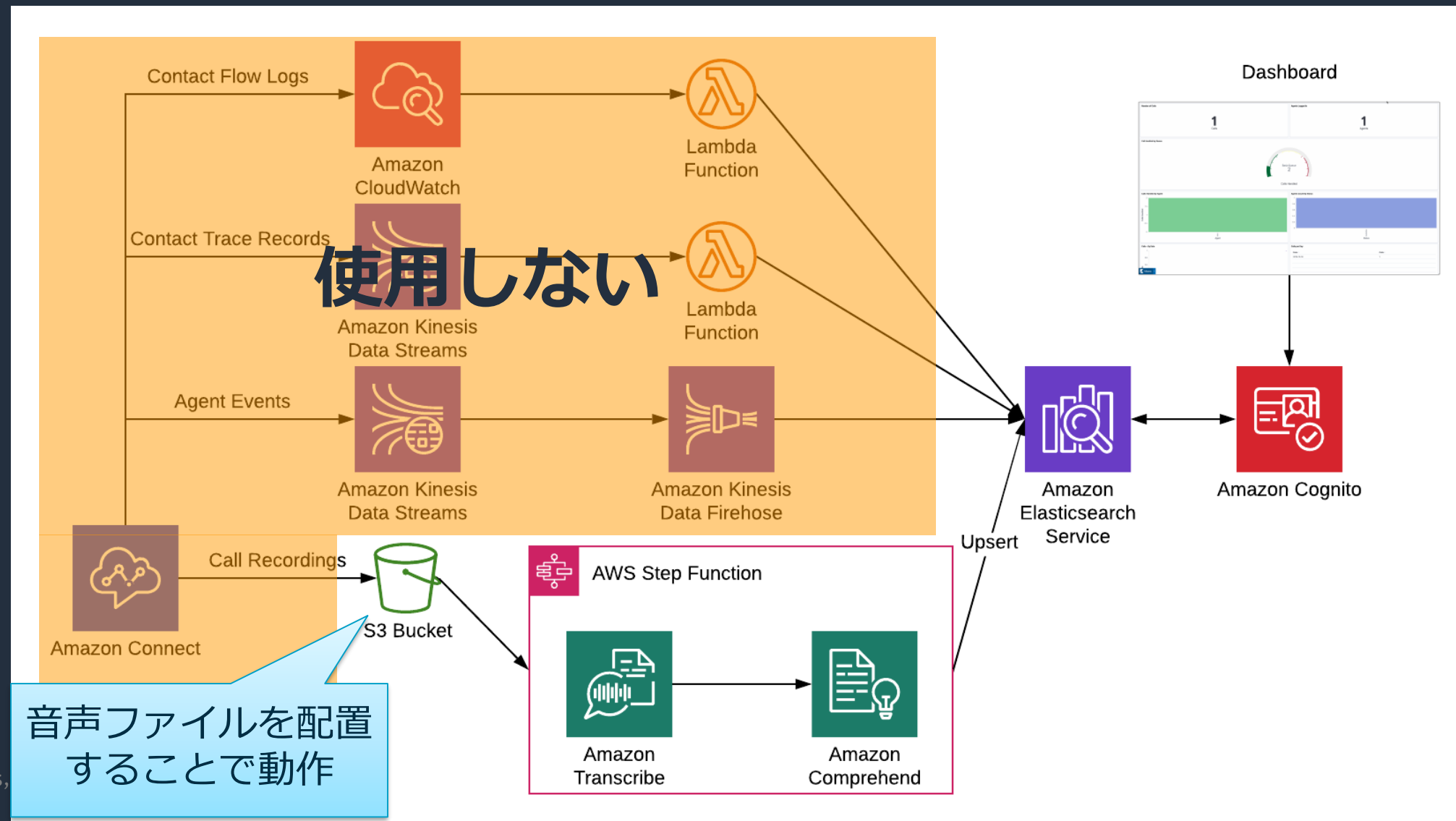
| Time ▾ | job_name | agent_name | customer_transcript |
|-------------------------------|----------|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| > Oct 30, 2020 @ 18:15:00.000 | M47LHN | gokazu | もしも数年前に買ったパソコンなんですけどちょうど一年だった時に電源が入らなくなったんですが壊れるタイマーで申し込んでいるんですかこんなひどい集中が初めてです無償修理願いたいです十一月日ですあそうなんですはいすぐをお願いしますはいこれをお願いしますお待ちしていればよろしいでしょうかありがとうございました |

- コール中の感情時系列トレンド (→から派生してセンテンス毎の分析を反映)



Voice of the customer integrations for handson

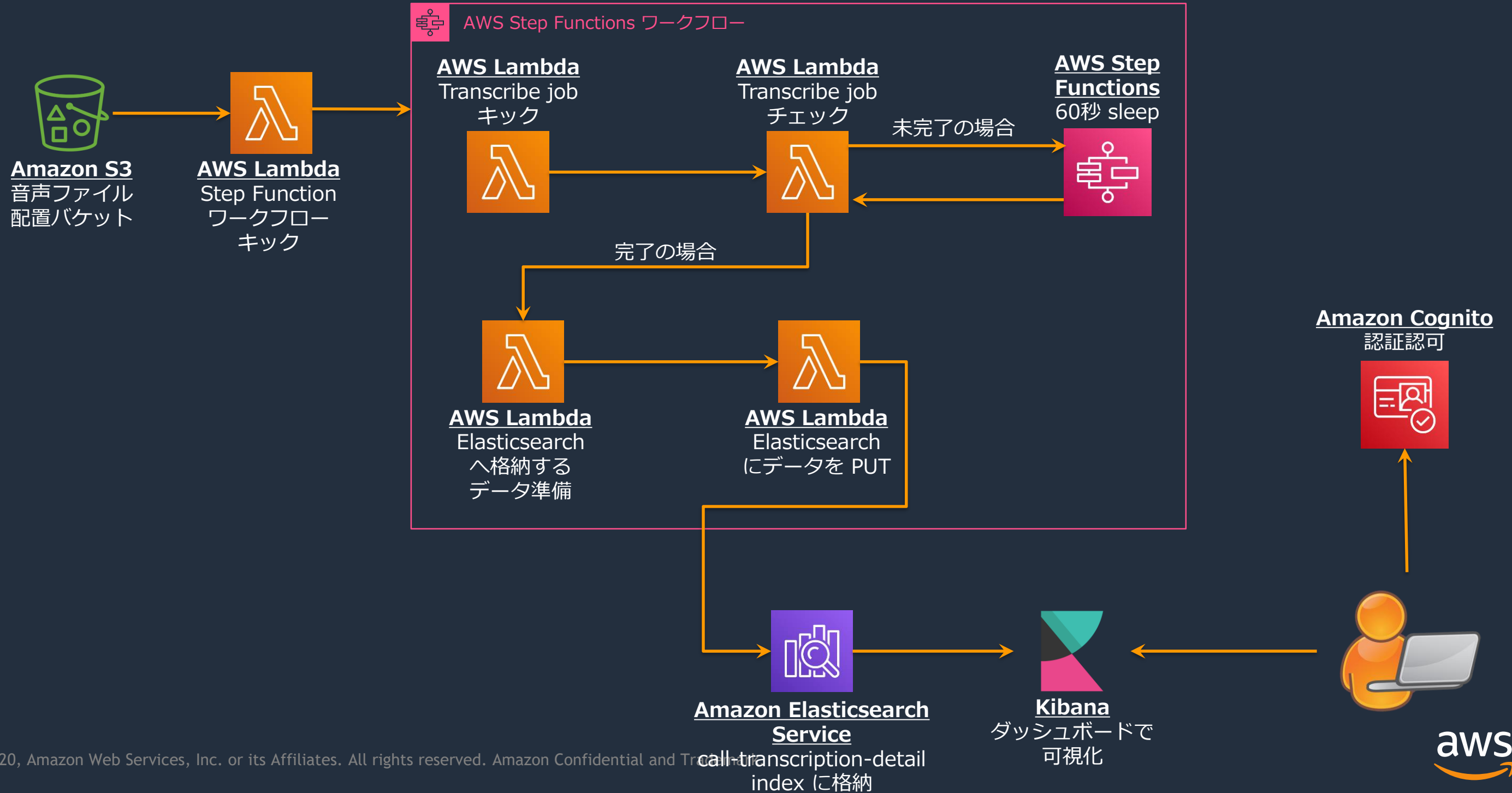
- <https://github.com/kazuhitogo/voice-of-customer-integration-handson>
- Amazon Connect 無しで動くように機能を縮小して準備



本ソリューションの利用目的例

- 録音データ活用による顧客満足度向上
 - 音声データを持っても有効活用は量が多く、全数検査は難しい
→ タグクラウドや感情推移から注視すべきコールを見つけやすい
 - コール全体でどんな話題が多いかが感覚的に把握できる
→ タグクラウドに浮かぶ単語の頻度から納期、品質保証などがわかる
- オペレーターの業務効率化
 - コールの後処理（サマリの記入）が負担大
→ 会話ログを表示することで記入の補助になる
 - マネージャによるエージェントのフォロー強化（問い合わせ品質、心理的フォロー）
→ 感情分析からフォローし易くなる

ソリューションのワークフロー



S3 音声ファイル配置バケット

バケットに対してファイル配置のトリガが設定され、.wav ファイルを配置することで発火して Lambda Function が動作する

Amazon S3

×

バケット

アクセスポイント

バッチオペレーション

S3 のアクセスアナライザー

ブロックパブリックアクセスのアカウント設定

注目機能 3

データイベントなし

表示するデータイベントがありません。

CloudTrail で設定する

イベント通知 (1)

バケットで特定のイベントが発生したときに通知を送信します。詳細はこちら

編集する

削除

イベント通知を作成

| <input type="checkbox"/> | 名前 | イベントタイプ | フィルター | 送信先タイプ | 送信先 |
|--------------------------|--------------------------------------|------------------|-------|-----------|-----------------------------------------------------------------------|
| <input type="checkbox"/> | 29d6529f-cc87-4ef8-85cd-39a81cbb3625 | すべてのオブジェクト作成イベント | , wav | Lambda 関数 | voc-test-kickOffStepFunctionsFromS3Event-KAURMAD5M0XK |

Lambda Function

S3 配置イベントで発火する Lambda Function

start_execution で Step Functions を起動

voc-test-kickOffStepFunctionsFromS3Event-KAURMAD5M0XK

スロットリング 限定条件 ▼ アクション ▼ テストイベントの選択 ▼ テスト

関数コード 情報 Deploy アクション ▼

File Edit Find View Go Tools Window Test Deploy

Environment

- ▼ voc-test-kickOffStep
- lambda_function.py

```
1 import json
2 import urllib.request, urllib.parse, urllib.error
3 import boto3
4 import os
5
6 print('Loading function')
7
8 s3 = boto3.client('s3')
9 stepfunctions = boto3.client('stepfunctions')
10
11 stepfunctionsarn = os.getenv('STEP_FUNCTIONS_ARN')
12
13 def lambda_handler(event, context):
14     # Get the object from the event and show its content type
15     bucket = event['Records'][0]['s3']['bucket']['name']
16     key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'])
17     try:
18         response = stepfunctions.start_execution(
19             stateMachineArn=stepfunctionsarn,
20             input=str(json.dumps({
21                 "dryrun": False,
22                 "audio_type": "audio/wav",
23                 "bucket": bucket,
24                 "key": key
25             })))
26     except Exception as e:
27         print(e)
28         print('Error getting object {} from bucket {}. Make sure they exist and your bucket is in the same region as this function.'.format(key, bucket))
29         raise e
30
31 return 'true'
```

1:1 Python Spaces: 4

Step Functions

音声データを文字に起こし、感情分析した結果を Amazon Elasticsearch に格納することまでの責任を持つ

編集 processAudioFile-662HAvx9fbFm 実行の開始 保存

変更により以前の値が上書きされます。実行では、開始時の定義が引き続き使用されます。

定義 エクスポート ▼ Layout ▼

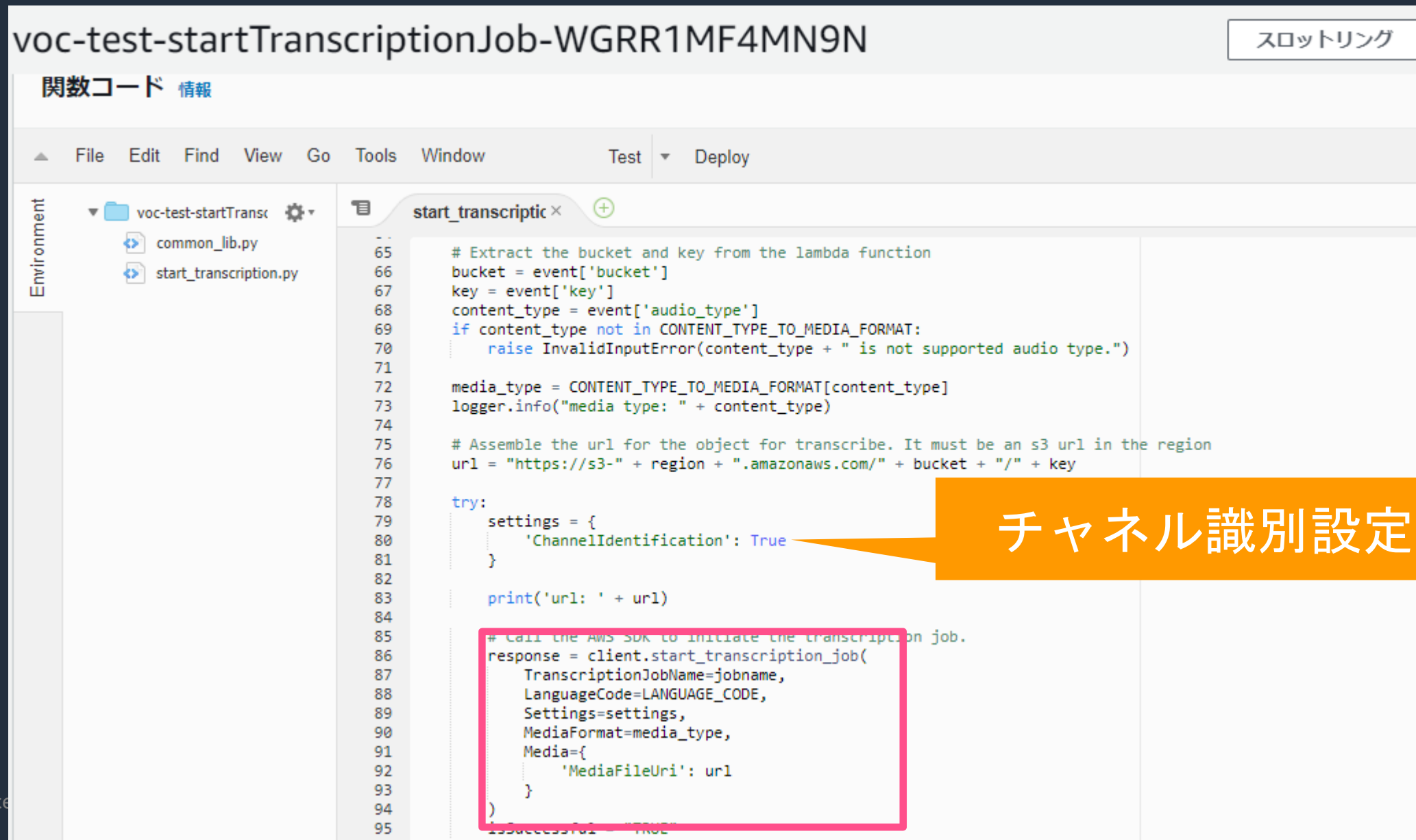
コードスニペットの生成 ▼ JSON のフォーマット

```
1 {
2   "StartAt": "Start Amazon Transcribe",
3   "States": {
4     "Start Amazon Transcribe": {
5       "Type": "Task",
6       "Resource": "arn:aws:lambda:ap-northeast-1:155580384669:function:voc-test-startTranscriptionJob-WGRR1MF4MN9N",
7       "InputPath": "$",
8       "ResultPath": "$.transcribe",
9       "Next": "Check Transcribe Status",
10      "Retry": [
11        {
12          "ErrorEquals": [ "ThrottlingException" ],
13          "IntervalSeconds": 120,
14          "BackoffRate": 1,
15          "MaxAttempts": 1000
16        },
17        {
18          "ErrorEquals": [ "States.ALL" ],
19          "IntervalSeconds": 60,
20          "BackoffRate": 2,
21          "MaxAttempts": 3
22        }
23      ]
24    }
25  }
```

```
graph TD
    Start((Start)) --> StartTranscribe[Start Amazon Transcribe]
    StartTranscribe --> CheckStatus[Check Transcribe Status]
    CheckStatus --> IsCompleted{Is Transcribe Completed?}
    IsCompleted -- No --> Wait[Wait for Transcribe Completion]
    Wait --> CheckStatus
    IsCompleted -- Yes --> GenerateTranscript[Generate Full Text Transcript]
    GenerateTranscript --> UploadElasticsearch[Upload To Elasticsearch]
    UploadElasticsearch --> Complete[Complete]
    Complete --> End((End))
```

Start Amazon Transcribe Lambda Function

start_transcription_job で文字起こしジョブを起動し、即座に完了する

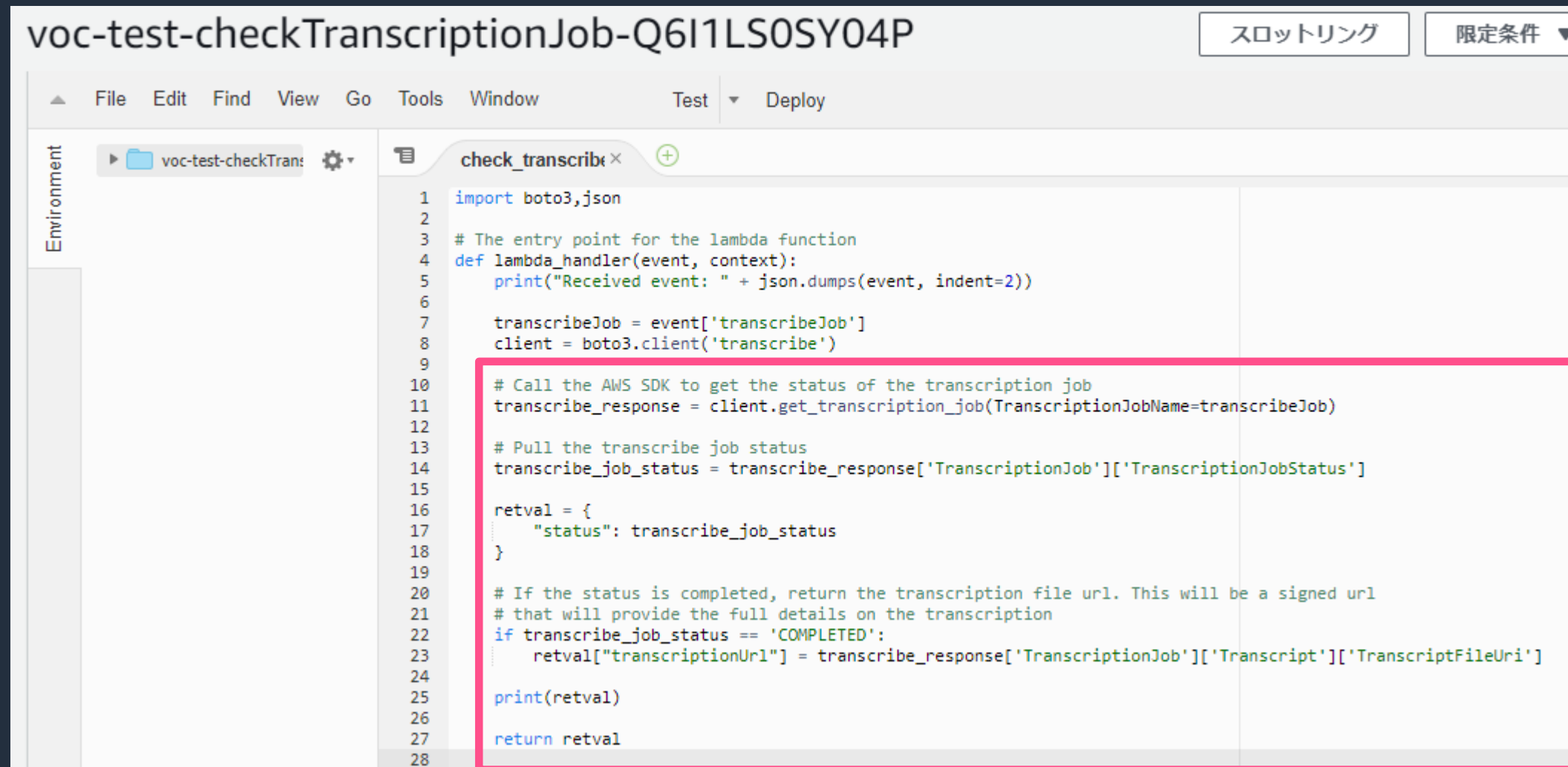


```
voc-test-startTranscriptionJob-WGRR1MF4MN9N スロットリング
関数コード 情報
File Edit Find View Go Tools Window Test Deploy
Environment
  ▼ voc-test-startTransc
    common_lib.py
    start_transcription.py
start_transcription.py
65 # Extract the bucket and key from the lambda function
66 bucket = event['bucket']
67 key = event['key']
68 content_type = event['audio_type']
69 if content_type not in CONTENT_TYPE_TO_MEDIA_FORMAT:
70     raise ValueError(content_type + " is not supported audio type.")
71
72 media_type = CONTENT_TYPE_TO_MEDIA_FORMAT[content_type]
73 logger.info("media type: " + content_type)
74
75 # Assemble the url for the object for transcribe. It must be an s3 url in the region
76 url = "https://s3-" + region + ".amazonaws.com/" + bucket + "/" + key
77
78 try:
79     settings = {
80         'ChannelIdentification': True
81     }
82
83     print('url: ' + url)
84
85     # Call the AWS SDK to initiate the transcription job.
86     response = client.start_transcription_job(
87         TranscriptionJobName=jobname,
88         LanguageCode=LANGUAGE_CODE,
89         Settings=settings,
90         MediaFormat=media_type,
91         Media={
92             'MediaFileUri': url
93         }
94     )
95     return response
```

チャンネル識別設定

Check Transcribe Status Lambda Function

文字起こしジョブが完了していたらそのURLを後段の Lambda Function のために連携する



```
1 import boto3,json
2
3 # The entry point for the lambda function
4 def lambda_handler(event, context):
5     print("Received event: " + json.dumps(event, indent=2))
6
7     transcribeJob = event['transcribeJob']
8     client = boto3.client('transcribe')
9
10    # Call the AWS SDK to get the status of the transcription job
11    transcribe_response = client.get_transcription_job(TranscriptionJobName=transcribeJob)
12
13    # Pull the transcribe job status
14    transcribe_job_status = transcribe_response['TranscriptionJob']['TranscriptionJobStatus']
15
16    retval = {
17        "status": transcribe_job_status
18    }
19
20    # If the status is completed, return the transcription file url. This will be a signed url
21    # that will provide the full details on the transcription
22    if transcribe_job_status == 'COMPLETED':
23        retval["transcriptionUrl"] = transcribe_response['TranscriptionJob']['Transcript']['TranscriptFileUri']
24
25    print(retval)
26
27    return retval
28
```

Generate Full Text Transcript Lambda Function

処理概要

- 1 秒以上沈黙があった場合は別のセンテンスに分ける
- それぞれのセンテンスと、全体それぞれにエンティティ、キーフレーズ、感情分析を行う
- 文字起こし結果を Python の dict 形式に格納して完了

(長いのでスクリーンショットは割愛)

Upload To Elasticsearch

前段の dict データを元に Elasticsearch へ格納する



The screenshot shows a web-based code editor interface. The title bar reads "voc-test-indexIntoElasticsearch-1DTXXIAC3IHGI". On the right side of the title bar, there are buttons for "スロットリング", "限定条件", "アクション", "テストイベントの選択", and "テスト". Below the title bar is a menu bar with "File", "Edit", "Find", "View", "Go", "Tools", "Window", "Test", and "Deploy". On the left side, there is an "Environment" panel showing a file tree with folders like "aws_requests_auth", "certifi", "chardet", "requests", and "urllib3". The main editor area has a tab titled "upload_to_elasti" and contains the following Python code:

```
59 def index_episode(es, event, fullEpisodeS3Location, contactId):
60     response = s3_client.get_object(Bucket=fullEpisodeS3Location['bucket'], Key=fullEpisodeS3Location['key'])
61     file_content = response['Body'].read().decode('utf-8')
62     fullepisode = json.loads(file_content)
63
64     s3_location = "s3://" + event['bucket'] + "/" + event['key']
65
66     s = event['key'].split('_')[1]
67
68     contact_id = contactId
69
70     updateDoc = {
71         'doc': {
72             'audio_type': event['audio_type'],
73             'audio_s3_location': s3_location,
74             'contact_id': contact_id,
75             'LastUpdateTimestamp': s[0:4] + '-' + s[4:6] + '-' + s[6:8] + 'T' + s.split('T')[1] + 'Z',
76             'type': 'CallRecord',
77             'wavfile_name': event['key'].split('/')[-1]
78         },
79         "doc_as_upsert" : True
80     }
81     for key in fullepisode.keys():
82         updateDoc['doc'][key] = fullepisode[key]
83
84     es.update(
85         index=SENTENCE_INDEX + '-' + s[0:4] + '-' + s[4:6] + '-' + s[6:8] if fullepisode['detail_flag'] == True else FULL_EPISODE_INDEX + '-' + s[0:4] + '-' + s[4:6] + '-' + s[6:8],
86         doc_type=FULL_EPISODE_DOCTYPE,
87         body=updateDoc,
88         id=contact_id
89     )
90
```

The code defines a function `index_episode` that takes `es`, `event`, `fullEpisodeS3Location`, and `contactId` as arguments. It retrieves a file from S3, loads it as JSON, and constructs an Elasticsearch document. The document is then indexed using `es.update`. The `updateDoc` dictionary is highlighted with a red box in the original image.

At the bottom right of the editor, it shows "32:1 Python Spaces: 4" and a gear icon.

データの格納結果

Kibana からクエリを投げて生データを確認することも可能
クエリ例) GET /detail-call-transcript-*/_search

Console

```
1 GET /detail-call-transcript*/_search
```

```
2 {  
3   "query": {  
4     "match_all": {}  
5   }  
6 }  
7
```

```
104 {  
105   "_index" : "detail-call-transcript-2020-09-30",  
106   "_type" : "doc",  
107   "_id" : "74c942b3-a614-4730-bf9f-85d3c5a3c6b9-00004",  
108   "_score" : 1.0,  
109   "_source" : {  
110     "audio_type" : "audio/wav",  
111     "audio_s3_location" : "s3://lpbucket-gokazu/connect/gokazu-voc2/CallRecordings/2020/09/30/74c942b3-a614-4730-bf9f-85d3c5a3c6b9_20200930T05:54.UTC.wav",  
112     "contact_id" : "74c942b3-a614-4730-bf9f-85d3c5a3c6b9-00004",  
113     "LastUpdateTimestamp" : "2020-09-30T05:54Z",  
114     "type" : "CallRecord",  
115     "wavfile_name" : "74c942b3-a614-4730-bf9f-85d3c5a3c6b9_20200930T05:54.UTC.wav",  
116     "job_name" : "RUHX4I",  
117     "person" : "agent",  
118     "start_time" : 7140,  
119     "end_time" : 14350,  
120     "content" : "畏まりもこう畏まりましたご迷惑をお掛けして申し訳ございません分かる状況を確認させていただきますコンセン트가入っておりますか",  
121     "agent_arn" : "arn:aws:connect:ap-northeast-1:155580384669:instance/d32ff148-302f-4adc-9b71-745148bbc58e/agent/12fd46f3-e4ac-49e6-8306-fb4d303e8f25",  
122     "agent_name" : "gokazu",  
123     "detail_flag" : true,  
124     "Positive" : 0.1341688632965088,  
125     "Negative" : 0.6766833662986755,  
126     "Neutral" : 0.18912507593631744,  
127     "Mixed" : 2.2685990188620053E-5,  
128     "sentiment" : "NEGATIVE",  
129     "KeyPhrases" : [  
130       "畏まり",  
131       "ご迷惑",  
132       "状況",  
133       "コンセンタ"  
134     ],  
135     "Entities" : [ ]  
136   }  
137 }
```



皆さまの今後の使用例

- 本ソリューションを皆さまの環境にデプロイいただく
- S3 に音声ファイルをアップロード
 - ドラッグ&ドロップによる手動作業が億劫な場合は AWS SDK や AWS CLI の利用をご検討ください
- 結果を確認して、ご利用価値がありそうか、応用できないかを検討

発展例

- Amazon Connect をすでにご利用の方はこちらのソリューションのご利用を検討ください
 - <https://github.com/kazuhitogo/voice-of-the-customer-integrations>
 - スタッフの名前や取った件数、Connect 自体やスタッフのログと連携することができます
- トピックモデリング / カスタム分類子の導入
- Amazon Forecast を用いてコール数を予測し、スタッフの配備計画に利用