Graphical user interface, application

Description automatically generated

**Coordinating AI Services with Step Functions**

**Introduction**

The individual machine learning services provided by AWS are incredibly powerful by themselves, but when used together, they are extraordinary. Chaining the services together can create truly magical experiences. However, the outputs and inputs of each of these services need to be coordinated because each service takes a varying amount of time based on the original input data. Step Functions are one way to keep track of all of these moving pieces.

In this lab, you will be modifying an existing pipeline to learn how to set up the coordination between the services. We will be using Lambda functions in the background to run the logic of our pipeline, but all of the Lambda functions are provided for you.

**Solution**

1. To avoid issues with the lab, use a new Incognito or Private browser window to log in to the lab. This ensures that your personal account credentials, which may be active in your main window, are not used for the lab.
2. Log in to the AWS console using the account credentials provided with the lab. Please make sure you are in the us-east-1 (N. Virginia) region when in the AWS console.
3. Download the example audio files that are provided for you on [GitHub](https://github.com/linuxacademy/content-aws-mls-c01/tree/master/CoordinateAIServicesWithStepFunctions).

**Add Translation to the Pipeline**

1. In the search bar, enter "lambda".
2. From the search results, select **Lambda**, and open it in a new tab.
3. Go back to the first tab, and enter "step functions" in the search bar.
4. From the search results, select **Step Functions**.
5. Click **Polyglot-Pipeline**.
6. Click **Edit**.
7. From *Generate a code snippet*, select **AWS Lambda: Invoke a function**.
8. From the *Lambda function* dropdown menu, select **Select function from a list**, and then select the **translate-lambda** function from the sub-menu.
9. Click **Copy to clipboard**.
10. Paste the code snippet on a new line under:
11. "End": true

},

1. Replace Invoke Lambda function with Translate Text.
2. Add a comma after the curly bracket after "NEXT\_STATE" and before "Transcribe Failed" so it looks like this:
3. "Next": "NEXT\_STATE"
4. },

"Transcribe Failed": {

**Add Sentiment Analysis to the Pipeline**

1. Go back to the Lambda functions tab.
2. Note the **comprehend-sentiment** function.
3. Go back to the Step Functions tab.
4. From *Generate a code snippet*, select **AWS Lambda: Invoke a function**.
5. From the *Lambda function* dropdown menu, select **Select function from a list**, and then select the **comprehend-sentiment** function from the sub-menu.
6. Click **Copy to clipboard**.
7. Paste the new function underneath the translate-lambda function.
8. Add a comma after the curly bracket after "NEXT\_STATE" and before "Transcribe Failed" so it looks like this:
9. "Next": "NEXT\_STATE"
10. },

"Transcribe Failed": {

1. Replace Invoke Lambda function with Comprehend Sentiment.
2. Under *Generate a code snippet*, select **Parallel state**.
3. Click **Copy to clipboard**.
4. Paste the snippet directly over the Transcript Available block.
5. Replace Parallel state with Transcript Available.
6. Delete the Pass State 3 block.
7. Copy the Translate Text block, delete it from its current position, and paste it over the "Pass State 1": { block.
8. For this block, replace "StartAt": "Pass State 1", with "StartArt": "Translate Text".
9. Also for this block, replace "Next": "NEXT\_STATE" with "End": true.
10. On the next line, delete the comma after the curly bracket. Scroll down, and below the Pass State 2 block, add a comma after the last curly bracket.
11. Above the code entry, click **Format JSON**.
12. Copy the Comprehend Sentiment block, delete it from its current position, and paste it over the "Pass State 2": { block.
13. Delete the comma after the last curly bracket in this block.
14. For this block, replace "StartAt": "Pass State 2", with "StartArt": "Comprehend Sentiment".
15. Also for this block, replace "Next": "NEXT\_STATE" with "End": true.
16. Above the code entry, click **Format JSON**.

**Convert the Translated Text to Audio**

1. Go back to the Lambda functions tab.
2. Note the **start-polly-lambda** function.
3. Go back to the Step Functions tab.
4. From *Generate a code snippet*, select **AWS Lambda: Invoke a function**.
5. From the *Lambda function* dropdown menu, select the **start-polly-lambda** function.
6. Click **Copy to clipboard**.
7. Paste the code snippet after:
8. "Next": "NEXT\_STATE"

}

1. Add a comma after the curly bracket after "NEXT\_STATE" and before "Transcribe Failed" so it looks like this:
2. "Next": "NEXT\_STATE"
3. },

"Transcribe Failed": {

1. Replace Invoke Lambda function with Convert Text to Speech.
2. Replace the preceding "Next": "NEXT\_STATE" with "Next": "Convert Text to Speech".
3. At the end of the block, replace "Next": "NEXT\_STATE" with "End": true.
4. Click **Format JSON**.
5. Click **Save**.
6. Click **Save anyway**.

After completing your steps, your State Machine Definition JSON should look something like this. Please notes, the <AWSAccountId> must match the account you're using for your lab:

{

"StartAt": "Start Transcribe",

"States": {

"Start Transcribe": {

"Type": "Task",

"Resource": "arn:aws:states:::lambda:invoke",

"Parameters": {

"FunctionName": "arn:aws:lambda:us-east-1:<AWSAccountId>:function:start-transcribe-lambda:$LATEST",

"Payload": {

"Input.$": "$"

}

},

"Next": "Wait for Transcribe"

},

"Wait for Transcribe": {

"Type": "Wait",

"Seconds": 45,

"Next": "Check Transcribe Status"

},

"Check Transcribe Status": {

"Type": "Task",

"Resource": "arn:aws:states:::lambda:invoke",

"Parameters": {

"FunctionName": "arn:aws:lambda:us-east-1:<AWSAccountId>:function:transcribe-status-lambda:$LATEST",

"Payload": {

"Input.$": "$"

}

},

"Next": "Is Transcribe Complete"

},

"Is Transcribe Complete": {

"Type": "Choice",

"Choices": [

{

"Variable": "$.Payload.TranscriptionJobStatus",

"StringEquals": "COMPLETED",

"Next": "Transcript Available"

},

{

"Variable": "$.Payload.TranscriptionJobStatus",

"StringEquals": "FAILED",

"Next": "Transcribe Failed"

}

],

"Default": "Wait for Transcribe"

},

"Transcript Available": {

"Type": "Parallel",

"Branches": [

{

"StartAt": "Translate Text",

"States": {

"Translate Text": {

"Type": "Task",

"Resource": "arn:aws:states:::lambda:invoke",

"Parameters": {

"FunctionName": "arn:aws:lambda:us-east-1:<AWSAccountId>:function:translate-lambda:$LATEST",

"Payload": {

"Input.$": "$"

}

},

"End": true

}

}

},

{

"StartAt": "Comprehend Sentiment",

"States": {

"Comprehend Sentiment": {

"Type": "Task",

"Resource": "arn:aws:states:::lambda:invoke",

"Parameters": {

"FunctionName": "arn:aws:lambda:us-east-1:<AWSAccountId>:function:comprehend-sentiment-lambda:$LATEST",

"Payload": {

"Input.$": "$"

}

},

"End": true

}

}

}

],

"Next": "Convert Text to Speech"

},

"Convert Text to Speech": {

"Type": "Task",

"Resource": "arn:aws:states:::lambda:invoke",

"Parameters": {

"FunctionName": "arn:aws:lambda:us-east-1:<AWSAccountId>:function:start-polly-lambda:$LATEST",

"Payload": {

"Input.$": "$"

}

},

"End": true

},

"Transcribe Failed": {

"Type": "Fail"

}

}

}

If your workflow diagram isn't quite matching the appearance in the Solution Video, ensure all of the "Next" and "End" directions in the workflow match the solution.

**Upload Audio and Watch the Magic**

1. In the search bar, enter "S3".
2. From the search results, select **S3**, and open it in a new tab.
3. Select the bucket whose name starts with **input**.
4. Click **Upload**.
5. Click **Add files**.
6. Select an audio file from your machine to upload, such as the GloomyDays.mp3 or GreatDayToBeYou.mp3 file provided for you in the GitHub repository for this lab, and click **Open**.
7. Click **Upload**.
8. Click the link to the bucket under *Destination*.
9. Check the audio file, and click **Actions**.
10. From the dropdown menu, select **Make public**.
11. Go back to the Step Functions tab.
12. Click the **Polyglot-Pipeline** link at the top.
13. Click the link under *Name*. The pipeline's status should now be *Running*.
14. View the **Step input** and **Step output** tabs to view the progress of the pipeline.
15. In the search bar at the top, enter "transcribe".
16. From the search results, select **Amazon Transcribe**, and open it in a new tab.
17. Click **Launch Amazon Transcribe**.
18. In the lefthand menu, click **Transcription jobs**.
19. Click the link under *Name*.
20. Under *Transcription preview*, view the translated text.
21. Go back to the Step Functions tab.
22. View the **Step input** and **Step output** tabs to view the progress of the pipeline. Select the **Comprehend Sentiment** box in the *Graph inspector*. In the output, you can view the sentiment of the text.
23. Go back to the S3 tab.
24. Click on the **Amazon S3** link on top.
25. Select the bucket whose name starts with **output**.
26. Click on the folder in the output folder, which should reflect the sentiment of the audio file.
27. Check the audio file and click **Actions**.
28. From the dropdown menu, click **Make public**.
29. Click **Make public**.
30. Click the link under *Source* to return to the bucket.
31. Click the link to the audio file.
32. Under *Object URL*, click the link to listen to the translated audio file.

**Conclusion**

Congratulations — you've completed this hands-on lab!