Introduction

AWS Step Functions is a powerful service that lets you build and orchestrate different AWS services to construct state machines. This can be done with almost any AWS API action, and with little-to-no code.

In this lab, we're going to build a Step Functions state machine to process an MP3 call recording, determine the sentiment of the conversation, and take a different action depending on the analysis.

Solution

Log in to the Lab Environment

1. To avoid issues with the lab, open 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 Management Console using the credentials provided on the lab instructions page. Make sure you're using the *us-east-1* region.

Create a Step Functions State Machine

1. In the search bar on top, type "Step Functions" to search for the Amazon Kinesis service.
2. Click on the **Step Functions** result to go directly to the Step Functions service.
3. Click on the **State machines** button in the sidebar.
4. Click on the **Create state machine** button to launch the wizard to create the state machine.
5. When prompted to select a template, opt for a blank state machine.
6. Navigate to the **Config** section of the state machine.
7. Update the **Execution role** to the pre-created `step-function-execution-role`, which will provide the necessary permissions for the lab.

Setup Amazon Transcribe

1. Add the `transcribe:StartTranscriptionJob` action to the start of the workflow.
2. Navigate to Amazon S3 in a new tab.
3. Open the bucket named `stepfunctionsbucket`.

4. Follow the following path to the GitHub Repository for this lab, and download the `conversation.mp3` file:
    - **Lab GitHub Repository**
5. Upload `conversation.mp3` to the S3 bucket.
6. Select the uploaded mp3 file to display the Properties page.
7. Copy the **S3 URI** for the uploaded file.
8. Return to the Step Functions tab.
9. Copy the following details to the *API Parameters* (replacing the necessary placeholders):

```
{
    "LanguageCode": "en-US",
    "Media": {
        "MediaFileUri": "<MP3-File-S3-URI>"
    },
    "TranscriptionJobName": "MyData",
    "OutputBucketName": "<S3-Bucket-Name>"
}
```

10. Rename the state name to **Start Call Transcription Job**.
11. Add the `transcribe:StartTranscriptionJob` action after the *Start Call Transcription Job* state.
12. Rename the state name to **Check Transcription Job**.
13. Add a **Choice** flow state after the *Check Transcription Job* state.
14. Set the conditions for the first Choice state rule to the following condition:
    - Variable: `$.TranscriptionJob.TranscriptionJobStatus`
    - Operator: **is equal to**
    - Value: **String constant**
    - `COMPLETED`
15. Add the comment `Transcript Completed` to the first rule.
16. Create a second rule for the Choice state.
17. Set the conditions for the first Choice state rule to the following condition:
    - Variable: `$.TranscriptionJob.TranscriptionJobStatus`
    - Operator: **is equal to**
    - Value: **String constant**
    - `IN_PROGRESS`

18. Add the comment `Transcript Processing` to the first rule.
19. Add a **Wait** flow state between the *Start Call Transcription Job* state, and the *Check Transcription Job* state.
20. Set the wait timer to a fixed interval of **2 seconds**.
21. Rename the state name to **Wait 2 seconds**.
22. Return to the **Choice** state.
23. Update the next state for the second rule (In Progress) to **Wait 2 seconds** to create a loop.
24. Rename the state name for the **Choice** state to **Transcript Status?**.
25. Add a **Success** flow state to the *Transcript Completed* branch of the **Transcript Status?** state.
26. Add a **Fail** flow state to the *Default* branch of the **Transcript Status?** state, in case of a failure.
27. Click on the **Create** button at the top-right of the screen to save the state machine.
28. Click on the **Start execution** button to trigger the first execution for our state machine.
29. Leave all of the contents of the execution window to the defaults, and click **Start execution**.
30. Wait for the job to be completed, and verify that it reaches the **Success** state.
31. Scroll to the top of the page, and click the **New execution** button to run the state machine again
32. Note the error due to the conflicting transcription job name, noting the requirements for a unique name.
33. Scroll to the top of the page, and click the **Edit state machine** button to return to the Workflow Studio.
34. Select the **Start Call Transcription Job** state, and update the *API Parameters* to the following (replacing the placeholders):

```
{
    "LanguageCode": "en-us",
    "Media": {
        "MediaFileUri": "<MP3-File-S3-URI>"
    },
    "TranscriptionJobName.$": "$$.Execution.Name",
    "OutputBucketName": "<S3-Bucket-Name>"
}
```

35. Select the **Check Transcription Job** state, and update the *API Parameters* to the following:

```
{
    "TranscriptionJobName.$": "$$.Execution.Name"
}
```

36. Click on the **Save** button at the top-right of the screen to save the state machine.
37. Click on the **Start execution** button to trigger the first execution for our state machine.
38. Leave all of the contents of the execution window to the defaults, and click **Start execution**.
39. Wait for the job to be completed, and verify that it reaches the **Success** state.

Setup Amazon Comprehend

1. Add the `comprehend:DetectSentiment` action in between the *Transcript Status?* state, and the *Success* state.
2. Add the `s3:GetObject` action in between the *Transcript Status?* state, and the *DetectSentiment* state.
3. Update the **GetObject** *API Parameters* to the following (replacing the placeholders):

```
{
    "Bucket": "<S3-Bucket-Name>",
    "Key.$": "States.Format('{}.json',
$.TranscriptionJob.TranscriptionJobName)"
}
```

4. Add a **Pass** flow state between the *GetObject* state, and the *DetectSentiment* state.
5. Configure the **Parameters** of the *Pass* state, and provide the following definition:

```
{
    "Body.$": "States.StringToJson($.Body)"
}
```

6. Select the **DetectSentiment** state, and update the *API Parameters* to the following:

```
{
    "LanguageCode": "en",
    "Text.$": "$.Body.results.transcripts[0].transcript"
}
```

7. Click on the **Save** button at the top-right of the screen to save the state machine.
8. Click on the **Start execution** button to trigger the first execution for our state machine.

9. Leave all of the contents of the execution window to the defaults, and click **Start execution**.
10. Wait for the job to be completed, and verify that it reaches the **Success** state.
11. Return to the Step Functions tab.
12. Rename the state name for the **GetObject** state to **Get Transcript File Contents**.
13. Rename the state name for the **Pass** state to **Parse the JSON**.
14. Rename the state name for the **DetectSentiment** state to **Detect Sentiment**.
15. Add a **Choice** flow state between the *Detect Sentiment* state, and the *Success* state.
16. Set the conditions for the first Choice state rule to the following condition:
    - Variable: `$.Sentiment`
    - Operator: **is equal to**
    - Value: **String constant**
    - `NEGATIVE`
17. Create a second rule for the Choice state.
18. Set the conditions for the first Choice state rule to the following condition:
    - Variable: `$.Sentiment`
    - Operator: **is equal to**
    - Value: **String constant**
    - `POSITIVE`
19. Add the `lambda:Invoke` action after the *Choice* state for the NEGATIVE sentiment path.
20. Configure the *Function Name* for the action to the `NegativeInteraction:$LATEST` function.
21. Set the *Next State* to **Success**.
22. Add the `sqs:SendMessage` action after the *Choice* state for the POSITIVE sentiment path.
23. Set the *Next State* to **Success**.
24. Configure the *Queue URL* for the action to the `PositiveInteractionQueue`.
25. Add a **Pass** state to the *Default* path for the *Choice* state.
26. Click on the **Save** button at the top-right of the screen to save the state machine.
27. Click on the **Start execution** button to trigger the first execution for our state machine.
28. Leave all of the contents of the execution window to the defaults, and click **Start**

28. Leave all of the contents of the execution window to the defaults, and click **Start execution**.
29. Wait for the job to be completed, and verify that it reaches the **Success** state.

Conclusion

Congratulations — you just learned how to create an AWS Step Function to process data and orchestrate events...