Coordinating AI Services with Step Functions (B)

Introduction

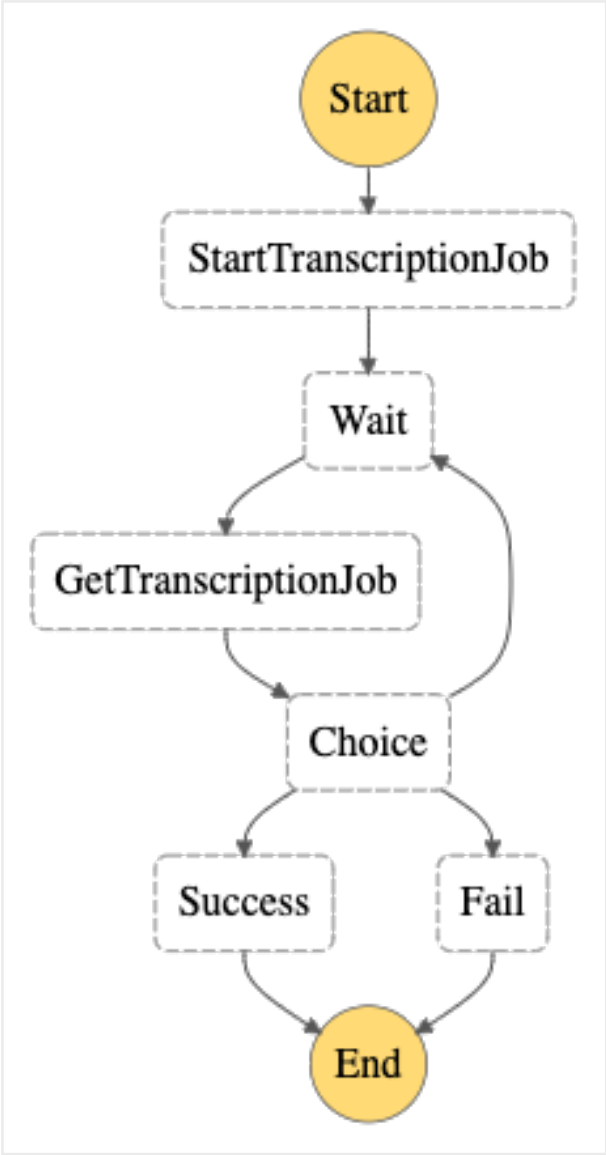
The individual machine learning services provided by AWS are incredibly powerful by themselves, but when used together, they are extraordinary. Chaining the services together can create truly magical experiences. However, the outputs and inputs of each of these services need to be coordinated because 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

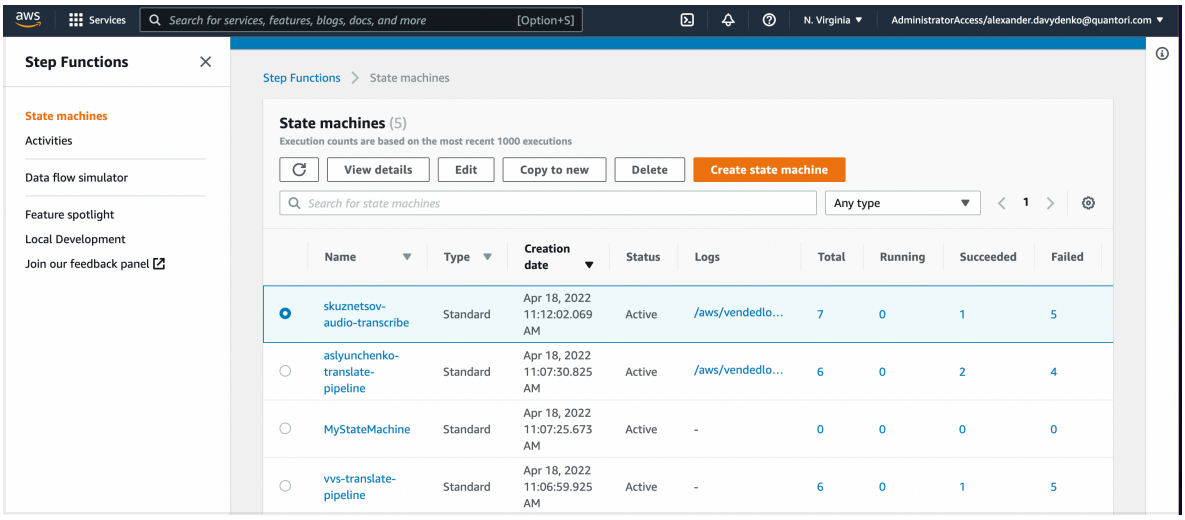
Download the example audio files that are provided for you on **GitHub**.

By this Lab we are creating a simple workflow and then enrich it with new capabilities:



Create new StepFunction:

1. Goto StepFunction Services
2. Click ‘Create state machine’



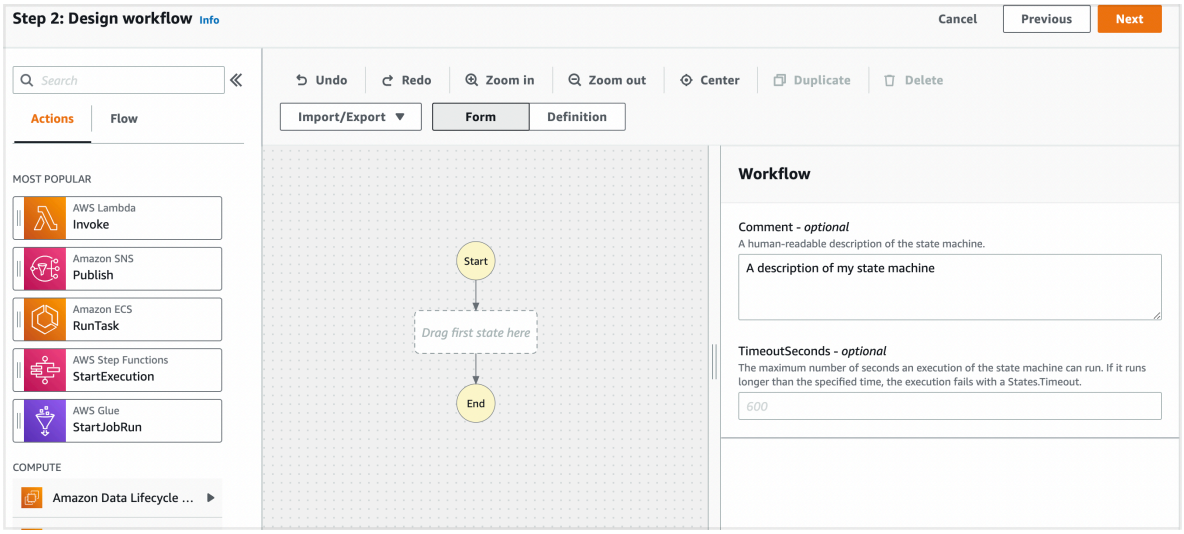
3. Define Step Function parameters:
    - «Design your workflow individually»
    - Type = Standard
    - «Next»

Transcribe Audio to Text

On this step we need to create *transcription* from the provided Audio file (located on the S3)

Start Transcription

1. As a result of a new Step Function creation an Editor should appear:



2. Add «StartTranscriptionJob» action as a very first step. It starts an asynchronous job to transcribe speech to text. Here we need to add parameter to initiate the Job:
    - ***Note:*** Individual states receive JSON as input from the previous state. If a state is first, it receives the execution input. A Task state can filter the JSON input, and parts of the input can be used as API parameters.

An example of the state input:

```
{
  "Bucket": "adavydenko-traslation-input",
  "Key": "liam.mp3",
  "S3Path": "s3://adavydenko-traslation-input/
liam.mp3",
  "JobId": "aaaa-bbbb-cccc"
}
```

- Specify API parameters according to the script. *TranscriptionJobName* must also be unique within an AWS account. If you try to create a transcription job with the same name as a previous transcription job, you get a ***ConflictException*** error. That's why we pass a synthetic Id, generated by lambda as an input for the StepFunction

```
{
  "IdentifyLanguage": true,
  "Media": {
    "MediaFileUri.$": "$.S3Path"
  },
  "TranscriptionJobName.$": "$.JobId"
}
```

Question: what could be an alternative for TranscriptionJobName?

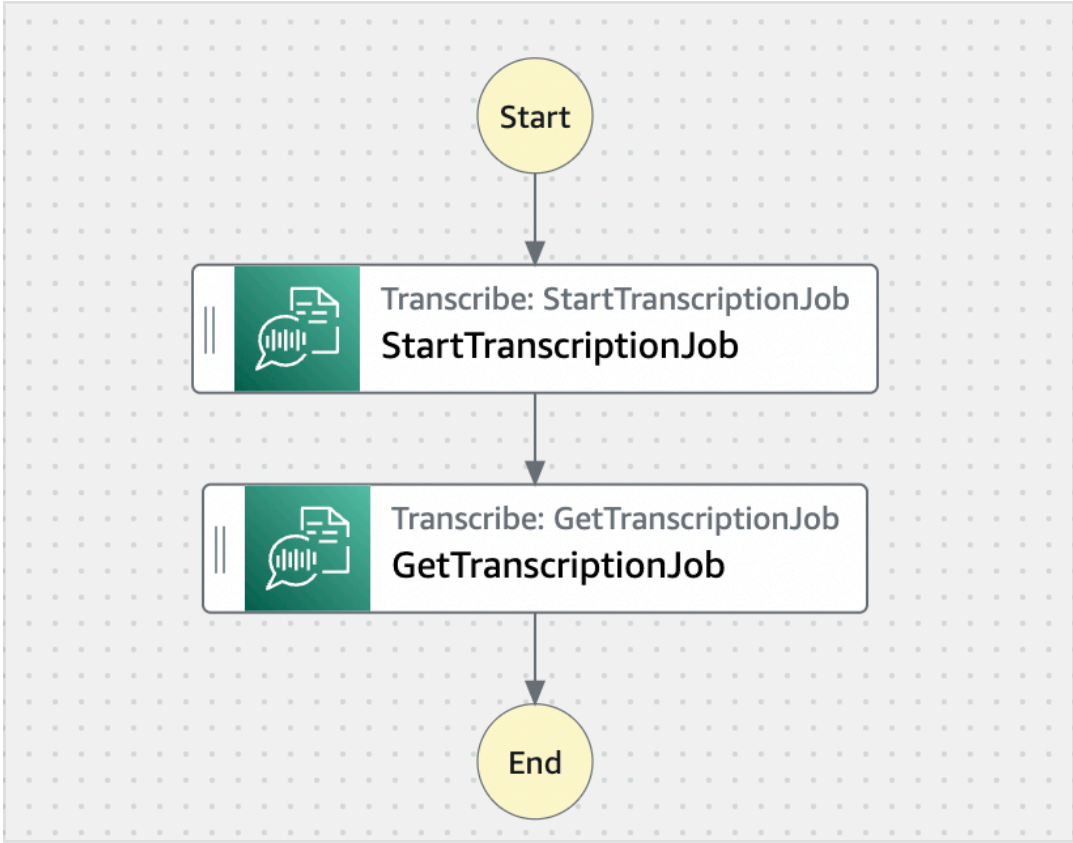
«StartTranscriptionJob» is an async action, it will return a TranscriptionJob object (https://docs.aws.amazon.com/transcribe/latest/APIReference/API_TranscriptionJob.html)

Wait for result of Transcription

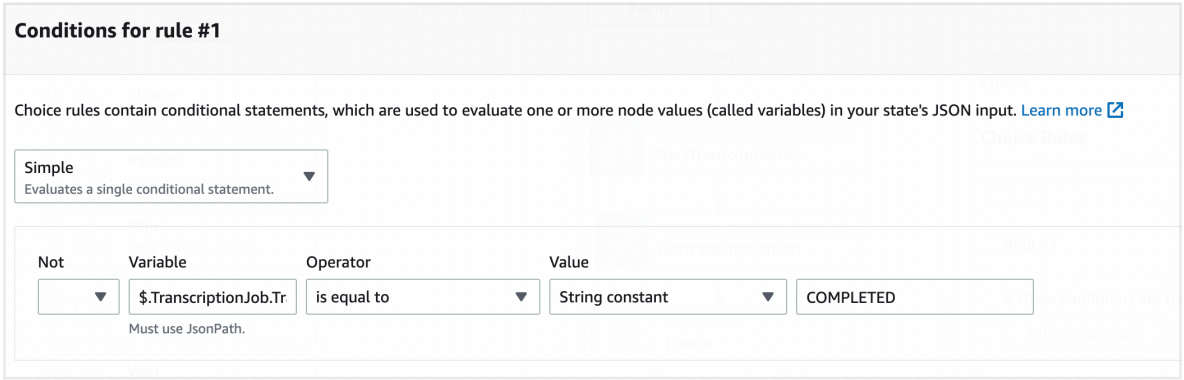
3. Add GetTranscriptionJob action.
   It returns information about a transcription job. To see the status of the job, check the *TranscriptionJobStatus* field. If the status is COMPLETED, the job is finished and you can find the results at the location specified in the *TranscriptFileUri* field.
   - Define API Parameters:

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

4. An intermediate workflow, should look like:

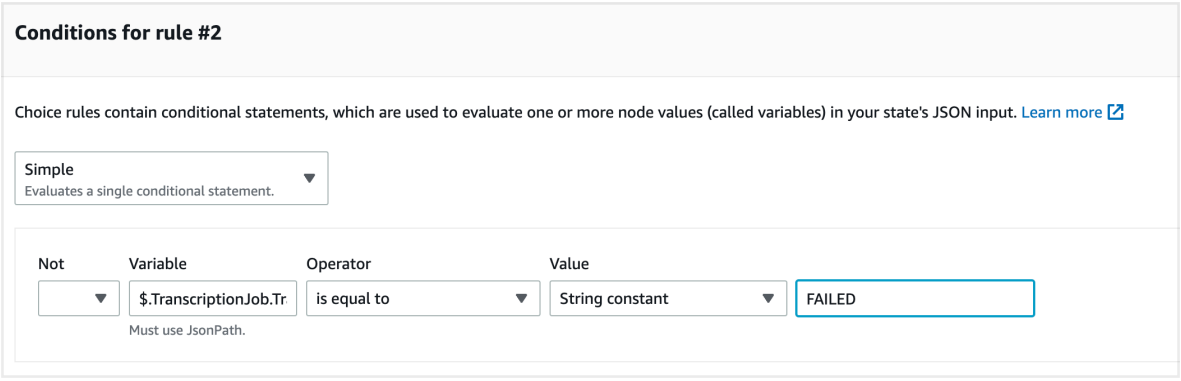


5. Transcription Job might not be completed immediately, it will take several seconds to complete the transcription, hence we need to wait for that and check status periodically (https://docs.aws.amazon.com/transcribe/latest/APIReference/API_TranscriptionJob.html#transcribe-Type-TranscriptionJob-TranscriptionJobStatus)
6. Add «Choice» flow
    - Check Completed status



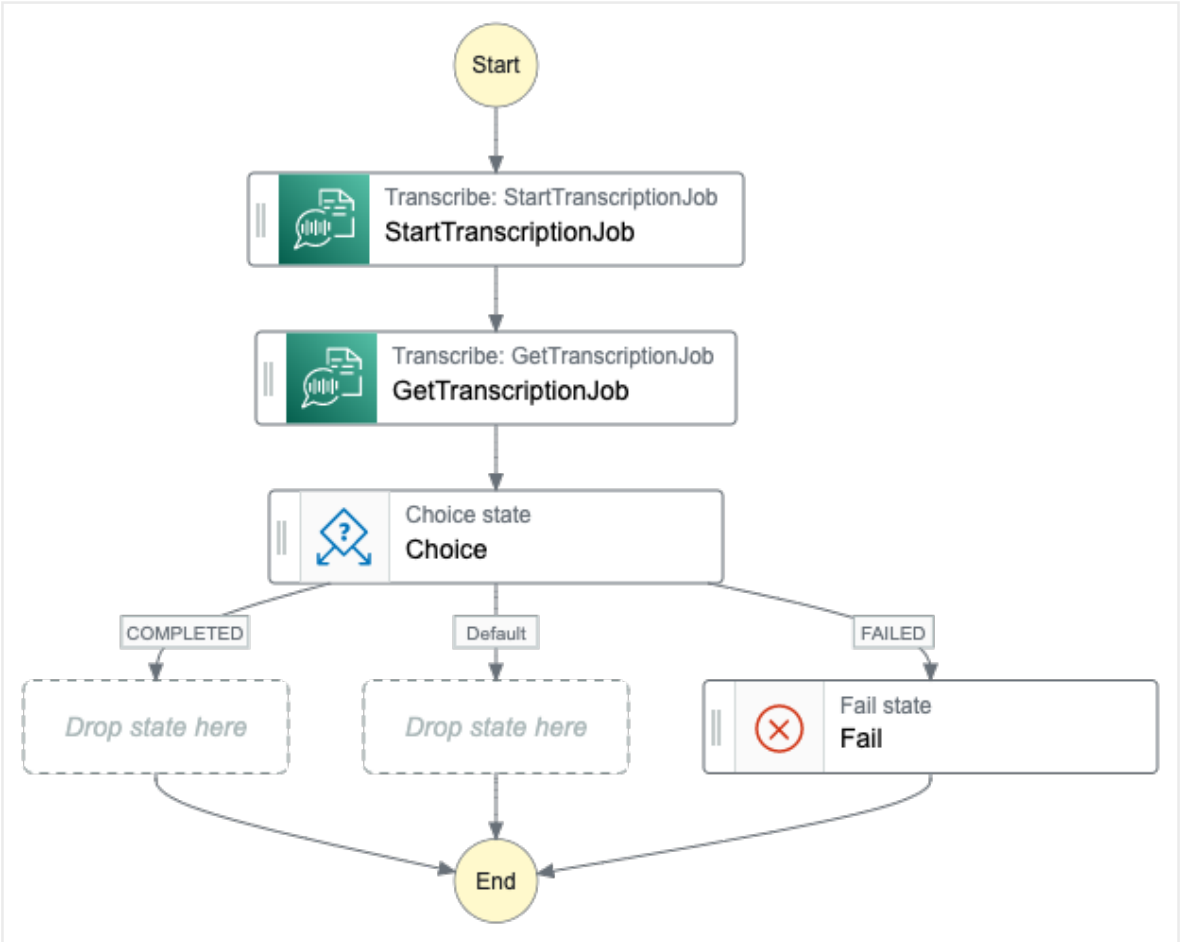
Condition: $.TranscriptionJob.TranscriptionJobStatus == "COMPLETED"

- Or Failed Status - Add new «Choice» rule and configure it:



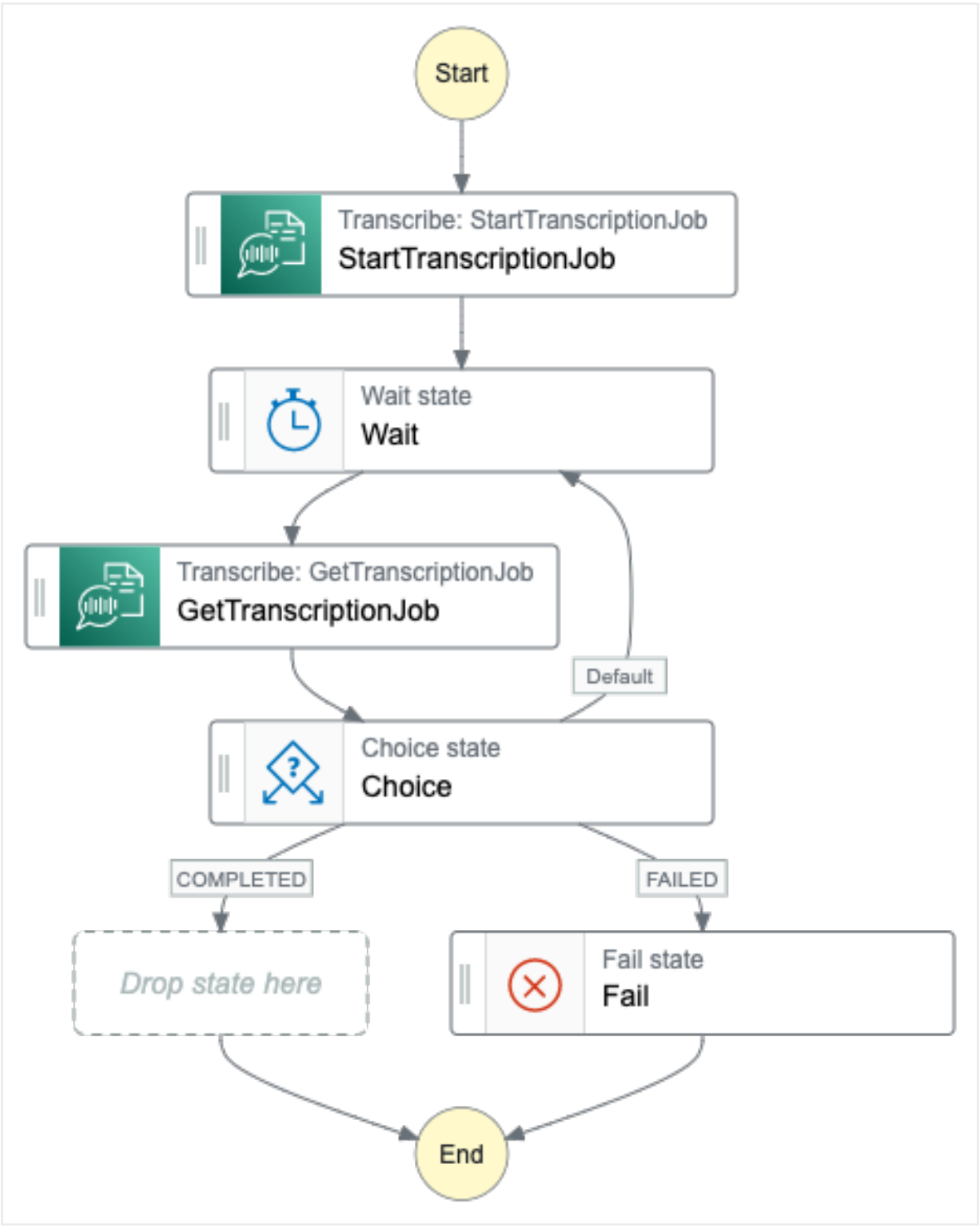
- For the FAILED branch add «Fail» flow as a resulting state for the given Choice
- By default - should wait

An intermediate result should look like:



7. Add «Wait» Flow before the «GetTranscriptionJob»
8. Link «Default» branch to the Wait State

The intermediate result should look like:



9. Click «Next» to review and create new state machine
    - Specify unique state machine name, e.g. «adavydenko-translate-pipeline»
    - Choose existing IAM role for pipeline or create a brand new one
10. Test pipeline by starting new execution an providing a json input, e.g.:

```
{
  "Bucket": "adavydenko-traslation-input",
  "Key": "liam.mp3",
  "S3Path": "s3://adavydenko-traslation-input/liam.mp3",
  "JobId": "5047b516-02ca-8e1f-5326-23fd63d2aceb"
}
```

11. Validate execution result, if it succeeded, you can find the output from the last State
12. Also navigate to the Amazon Transcribe service, locate your Job by Id and validate result

Part I Definition:

translate-pipeliene-managed-1.asl.json
JSON · 3 КБ

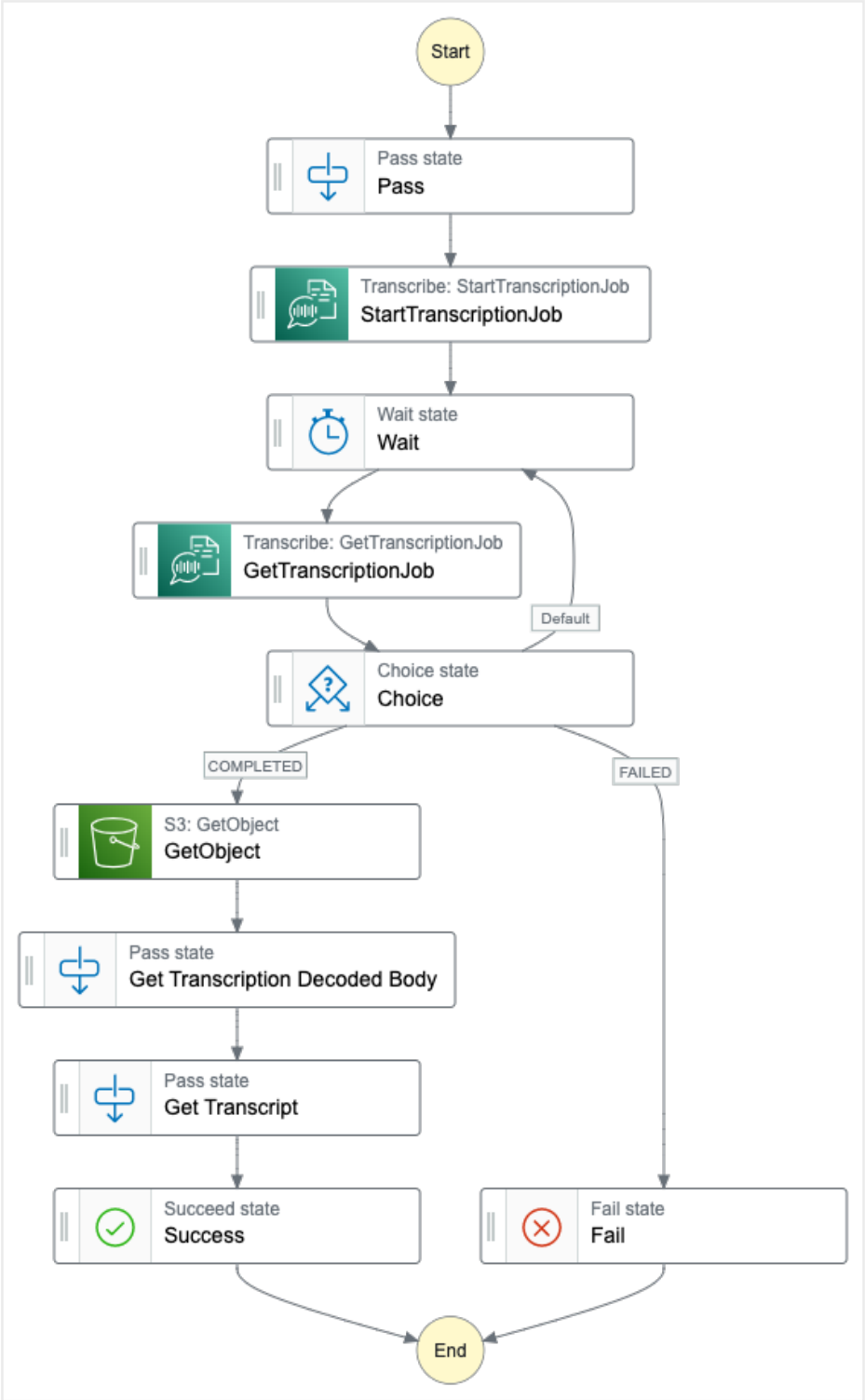
Copy results of transcription

By default, *StartTranscriptionJob* will store results under Service-Managed S3 Bucket. *OutputBucketName* controls that behavior.

- If you set the OutputBucketName, Amazon Transcribe puts the transcript in the specified S3 bucket. When you call the GetTranscriptionJob operation, the operation returns this location in the TranscriptFileUri field. The S3 bucket must have permissions that allow Amazon Transcribe to put files in the bucket.
- If you don't set the OutputBucketName, Amazon Transcribe generates a pre-signed URL, a shareable URL that provides secure access to your transcription, and returns it in the TranscriptFileUri field. Use this URL to download the transcription.

13. Create new «Pass» state which will modify input and define new variables:
14. Don’t forget to check IAM Role of the Step Function - so that it has write access to your bucket
15. Get Results from S3 from Transcription Job & Decode them

Part II Definition



translate-pipeliene-managed-2.asl.json
JSON · 1 КБ

Add Translation to the Pipeline
[SELF]

Add Sentiment Analysis to the Pipeline
[SELF]

Convert the Translated Text to Audio
[SELF]

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