



HELLO!

STEP INTO SERVERLESS WITH AWS & CFML

Brian Klaas
brian.klaas@gmail.com
@brian_klaas



HI, LAMBDA!

YES, THERE ARE STILL SERVERS...



**NO SERVER
IS EASIER TO MANAGE
THAN NO SERVER.**

SERVERLESS IS APPLICATION FUNCTIONALITY

STORAGE

S3, GLACIER, EFS

MEDIA

ELASTIC TRANSCODER, MEDIA CONVERT,
MEDIA LIVE, KINESIS VIDEO STREAMS

AI

REKOGNITION, POLLY, TRANSLATE,
TRANSCRIBE, COMPREHEND, LEX,
SAGEMAKER

DB

RDS, DYNAMODB, AURORA, REDSHIFT

CONTAINER

ECS, FARGATE



SERVERLESS = FUNCTIONS

HELLO LAMBDA (NODE.JS)

```
exports.handler = (event, context, callback) => {  
  var firstNumber = event.firstVal;  
  var secondNumber = event.secondVal;  
  var result = firstNumber * secondNumber;  
  callback(null, result);  
};
```



(RE)BUILDING THE MICROLITH

VIDEO TRANSCODING WORKFLOW

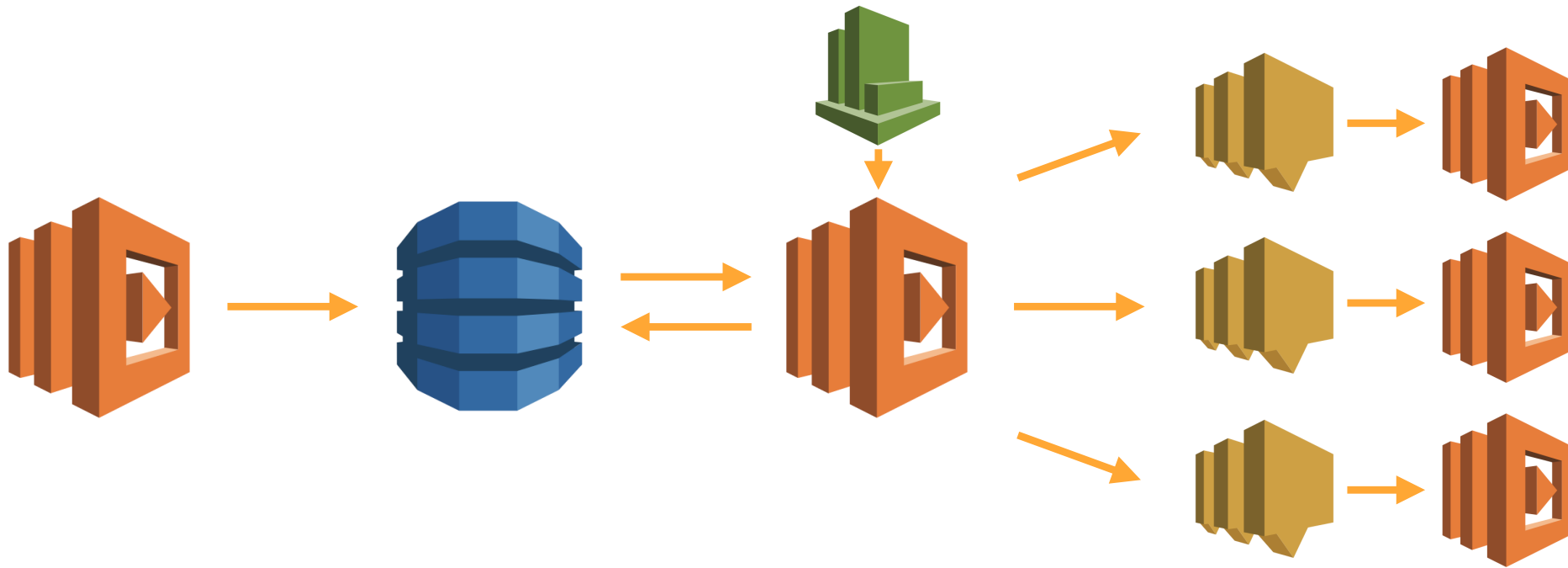




THE PROBLEM

**WE DON'T BUILD FUNCTIONS.
WE BUILD APPLICATIONS.**

ORCHESTRATE WORKFLOWS BETWEEN FUNCTIONS?





THE PROBLEM

**BRANCHING?
ERROR HANDLING?
RETRIES?**



STEP FUNCTIONS

PROBLEM SOLVED!

EMBRACE AWS EXTEND WITH CFML

EMBRACE AI SERVICES IN AWS EXTEND WITH CFML

AWS Service Playbox

CFML

[Lambda Function Invocation](#)

[DynamoDB](#)

[Simple Notification Service \(SNS\)](#)

[Relational Database](#)

AWS PLAYBOX APP

<https://github.com/brianklaas/awsPlaybox>



SORRY, NO IAM



WHAT ARE STEP FUNCTIONS?

WHAT ARE STEP FUNCTIONS?



WHAT ARE STEP FUNCTIONS?

VISUAL, SERVERLESS ORCHESTRATION



WHAT ARE STEP FUNCTIONS?

AUTOMATED, MULTI-STEP, ASYNCHRONOUS, SERVERLESS WORKFLOWS IN AWS

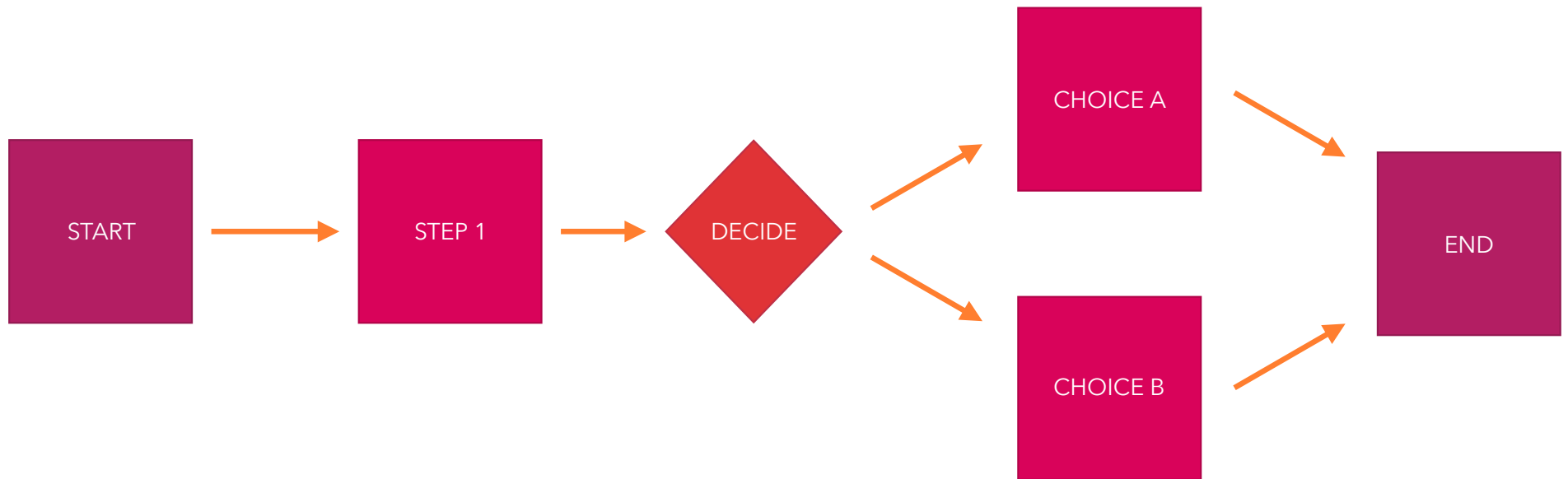


WHAT ARE STEP FUNCTIONS?

STATE MACHINES



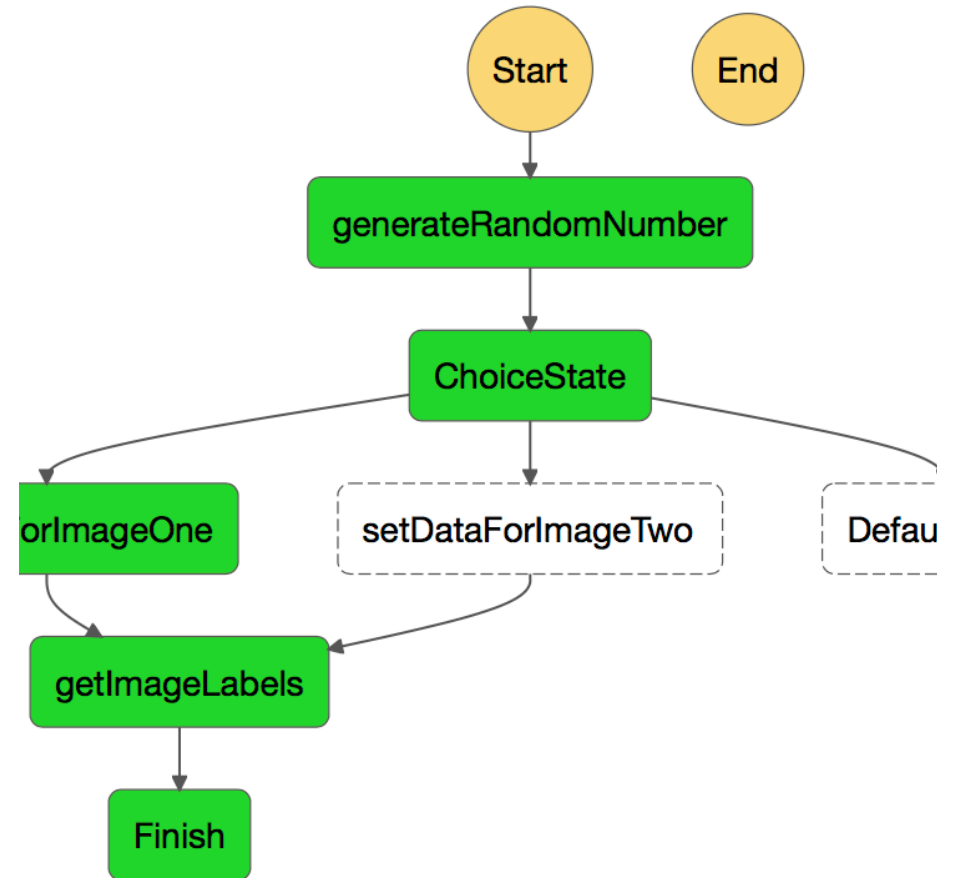
WHAT DOES STATE-BASED WORKFLOW LOOK LIKE?





VISUAL ORCHESTRATION

- Via the AWS Console





STATE TYPES TO CHOOSE FROM

- Task
- Sequence
- Parallel
- Branch
- Wait
- Success
- Failure



SHARE DATA BETWEEN STEPS

- Must be expressed in JSON
- Traverse with JSONPath pathing (<https://github.com/json-path/JsonPath>)
- Reference path: \$.someVariableName
- InputPath = filter selection of current state's raw input to go into task (function)
- ResultPath = reference path to data coming out of task (function)
- OutputPath = filter selection of current task (function) result, to serve as input for next state



BUILD AND VALIDATE STATE MACHINES IN THE AWS CONSOLE

- Not using the console? Use statelint
<https://github.com/awslabs/statelint>

Changes will overwrite previous values. Running executions will continue to use the definition they w

IAM role for executions

StatesExecutionRole-us-east-1



Create new role

Code

```
1 {
2   "Comment": "A simple example of making choices in Step
3   Functions.",
4   "StartAt": "generateRandomNumber",
5   "States": {
6     "generateRandomNumber": {
7       "Type": "Task",
8       "Resource": "arn:aws:lambda:us-east-
9       :function:confDemoRandomNumber",
10      "ResultPath": "$.randomNumber",
11      "Next": "ChoiceState"
12    },
13    "ChoiceState": {
14      "Type": "Choice",
15      "Choices": [
16        {
17          "Variable": "$.randomNumber",
18          "NumericLessThanEquals": 50,
19          "Next": "setDataForImageOne"
20        },
21        {
22          "Variable": "$.randomNumber",
23          "NumericGreaterThan": 50,
24          "Next": "setDataForImageTwo"
25        }
26      ]
27    }
28  }
29 }
```

Vis

set

EXAMPLE WORKFLOW: DESCRIBE AN IMAGE



LAMBDA



S3



REKOGNITION



CFML!

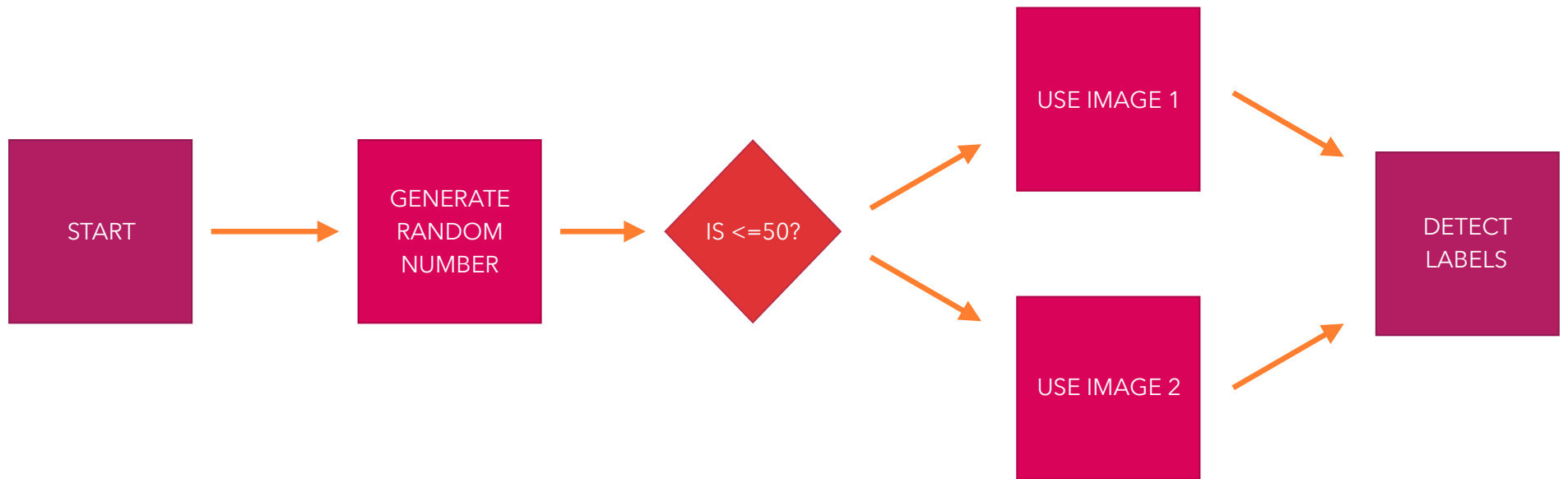


STATE TYPES IN THE EXAMPLE

- ◆ Task
- ◆ Choice
- ◆ Pass
- ◆ Fail
- ◆ Succeed

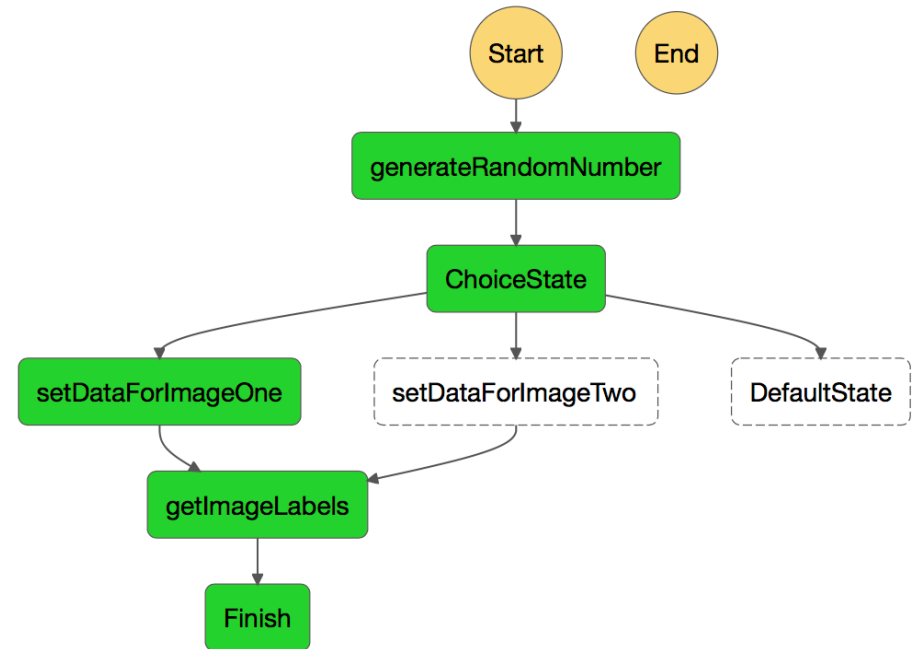


WORKFLOW OVERVIEW





EXECUTION DIAGRAM

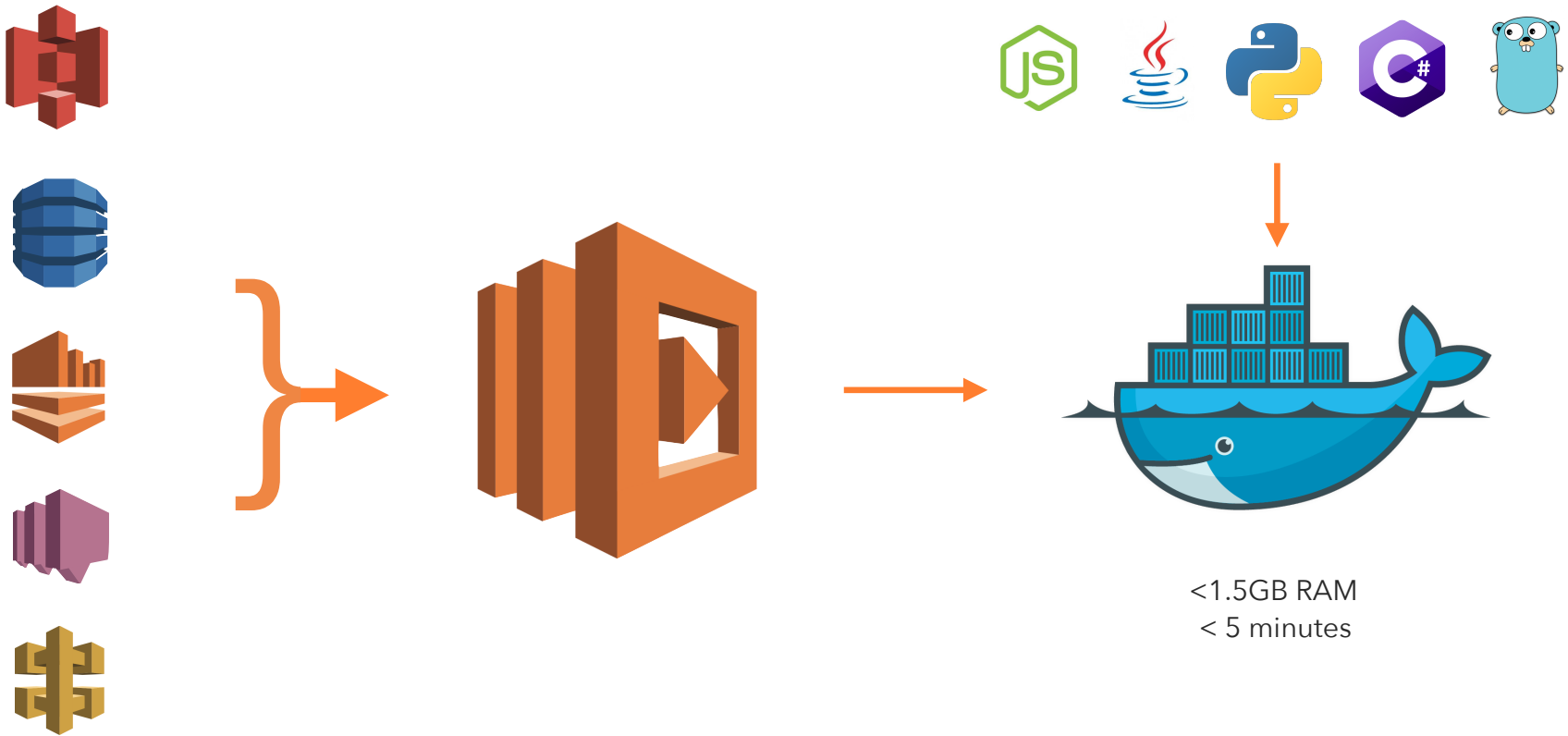




TASK STATES

```
"States": {  
  "generateRandomNumber": {  
    "Type": "Task",  
    "Resource": "arn:aws:lambda:us-east-1:XXXXXXXXX:function:generateRandomNumber",  
    "ResultPath": "$.randomNumber",  
    "Next": "ChoiceState"  
  },  
}
```

HOW DOES LAMBDA WORK?





WRITING CODE FOR LAMBDA

- Upload a ZIP containing source and dependencies
 - Must zip the folder contents, not the containing folder
 - 50MB size limit
- Point to a ZIP on S3
- Code using Cloud9 Editor in the console



```
File Edit Find View Goto Tools Window
cfDemoConvertTextToSpeec
index.js
index.js
1 // As speaking jobs can take a few seconds, the default Lambda function
2
3 const util = require('util');
4 const AWS = require('aws-sdk');
5 const Polly = new AWS.Polly();
6 const S3 = new AWS.S3();
7
8 exports.handler = (event, context, callback) => {
9     console.log("Reading input from event:\n", util.inspect(event, {dept
10
11     var textToSpeak = event.textToSpeak;
12     var languageOfText = event.languageOfText;
13     var fileNameForOutput = event.transcriptFileName;
14
15     // We have to use promises for both steps in the process because S3
16     makeMP3FromText(textToSpeak, languageOfText).then(function(makeMP3Re
17         console.log("Result from speaking transcript:", makeMP3Result);
18         return writeFileToS3(makeMP3Result.AudioStream, languageOfText,
19     }).then(function(mp3FileNameOnS3) {
20         console.log("mp3FileNameOnS3:" + mp3FileNameOnS3);
21         callback(null, mp3FileNameOnS3);
22     }).catch(function(err) {
23         console.error("Failed to generate MP3!", err);
24         callback(err, null);
25     })
26 };
27
28 function makeMP3FromText(textToSpeak, languageOfText) {
29     return new Promise(function(resolve, reject) {
30         console.log("Making an MP3 in the language: " + languageOfText);
31         var voiceToUse = 'Ivy';
32         // Polly has a current maximum character length of 3000 characte
33         var maxLength = 3000;
```



GENERATE RANDOM NUMBER FUNCTION

```
exports.handler = (event, context, callback) => {  
  var min = event.min ? event.min : 1;  
  var max = event.max ? event.max : 100;  
  var result = Math.floor(Math.random() * (max - min)) + min;  
  callback(null, result);  
};
```



TASK STATES

```
"States": {  
  "generateRandomNumber": {  
    "Type": "Task",  
    "Resource": "arn:aws:lambda:us-east-1:XXXXXXXXXXXX:function:generateRandomNumber",  
    "ResultPath": "$.randomNumber",  
    "Next": "ChoiceState"  
  },  
}
```



CHOICE STATES

```
"ChoiceState": {
  "Type": "Choice",
  "Choices": [
    {
      "Variable": "$.randomNumber",
      "NumericLessThanEquals": 50,
      "Next": "setDataForImageOne"
    },
    {
      "Variable": "$.randomNumber",
      "NumericGreaterThan": 50,
      "Next": "setDataForImageTwo"
    }
  ],
  "Default": "WhyAreWeHereState"
```



COMPARATORS FOR CHOICE STATES

StringEquals

StringLessThan

StringGreaterThan

StringLessThanEquals

StringGreaterThanEquals

NumericEquals

NumericLessThan

NumericGreaterThan

NumericLessThanEquals

NumericGreaterThanEquals

BooleanEquals

TimestampEquals

TimestampLessThan

TimestampGreaterThan

TimestampLessThanEquals

TimestampGreaterThanEquals

And

Or

Not



CHOICE STATES

```
"ChoiceState": {  
  "Type": "Choice",  
  "Choices": [  
    {  
      "Variable": "$.randomNumber",  
      "NumericLessThanEquals": 50,  
      "Next": "setDataForImageOne"  
    },  
    {  
      "Variable": "$.randomNumber",  
      "NumericGreaterThan": 50,  
      "Next": "setDataForImageTwo"  
    }  
  ],  
  "Default": "WhyAreWeHereState"
```




PASS STATES

```
"setDataForImageOne": {  
  "Type" : "Pass",  
  "Result": { "s3Bucket": "conferencedemobucket", "s3Key": "images/familyFaces.jpg" },  
  "Next": "getImageLabels"  
},
```

```
"setDataForImageTwo": {  
  "Type" : "Pass",  
  "Result": { "s3Bucket": "conferencedemobucket", "s3Key": "images/dogForLabels.jpg" },  
  "Next": "getImageLabels"  
},
```



REKOGNIZING THE IMAGE

```
"getImageLabels": {  
  "Type" : "Task",  
  "Resource": "arn:aws:lambda:us-east-1:XXXXXXXXX:function:detectLabelsForImage",  
  "Next": "Finish"  
},
```



EXAMPLE 1: DESCRIBING AN IMAGE



REKOGNITION



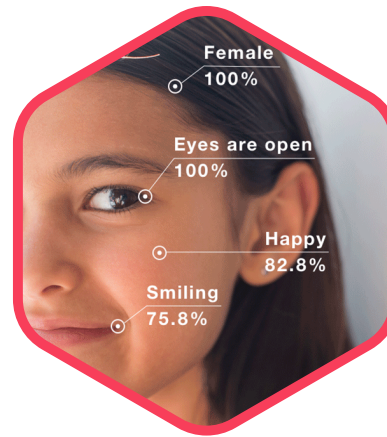
WHAT CAN REKOGNITION DO?



DETECT IMAGE ELEMENTS



IDENTIFY FACES



ANALYZE/MATCH FACES



IDENTIFY TEXT



DETECT LABELS FOR IMAGE (NODE.JS)

```
var params = {  
  Image: {  
    S3Object: {  
      Bucket: srcBucket,  
      Name: srcKey  
    }  
  },  
  MaxLabels: 10,  
  MinConfidence: 70  
};  
  
reognition.detectLabels(params)  
.promise().then(function (data) {  
  data.s3Bucket = srcBucket;  
  data.s3Key = srcKey;  
  callback(null, data);  
}).catch(function (err) {  
  callback(err);  
});
```

Ho



EXAMPLE 1: DESCRIBING AN IMAGE

WHAT DOES RECOGNITION TELL US?



Here are the labels:

- Architecture — 90%
- Building — 90%
- Castle — 90%
- House — 90%
- Housing — 90%
- Mansion — 90%
- Palace — 90%
- Gazebo — 84%
- Altar — 82%
- Pagoda — 57%
- Shrine — 57%
- Temple — 57%
- Worship — 57%
- Monastery — 52%
- Church — 51%
- Cathedral — 51%



END STATES

```
"WhyAreWeHereState": {  
  "Type": "Fail",  
  "Cause": "We really should not have ended up here from a numeric value decision."  
},  
  
"Finish": {  
  "Type": "Succeed"  
}
```



EXAMPLE 1: DESCRIBING AN IMAGE

SO WHERE DOES **CF** COME IN?



CF IS THE SCHEDULER AND EXECUTOR



YOUR CFML NEEDS TO:

- Make an execution request
- Get the ARN of the execution for follow-up
- Check on completion
- If complete, get the results



USING THE AWS JAVA SDK

- Add to cfusion/lib:
 - aws-java-sdk-1.11.xxx.jar
 - jackson-annotations-2.6.0.jar
 - jackson-core-2.6.7.jar
 - jackson-databind-2.6.7.1.jar
 - joda-time-2.8.1.jar





MAKE AN EXECUTION REQUEST

```
arnOfFunctionToInvoke = "arn:aws:states:us-  
east-1:XXXXXXXXXX:stateMachine:confDemoSimpleDecisionMachine"
```

```
executionRequest = CreateObject('java',  
'com.amazonaws.services.stepfunctions.model.StartExecutionRequest').init();
```

```
executionRequest.setStateMachineArn(arnOfFunctionToInvoke);
```

```
executionResult = stepFunctionService.StartExecution(executionRequest);
```



GET ARN OF EXECUTION

```
executionResult = stepFunctionService.StartExecution(executionRequest);
```

```
executionARN = executionResult.getExecutionARN();
```



CHECK ON COMPLETION

```
describeExecutionRequest = CreateObject('java',  
'com.amazonaws.services.stepfunctions.model.DescribeExecutionRequest').init();  
  
describeExecutionRequest.setExecutionArn(executionARN);  
  
executionResult = stepFunctionService.describeExecution(describeExecutionRequest);
```



IF COMPLETE, GET THE RESULTS

```
stepFunctionResult.status = executionResult.getStatus();  
  
if (stepFunctionResult.status IS 'SUCCEEDED') {  
  
    stepFunctionResult.finishedOn = executionResult.getStopDate();  
  
    stepFunctionResult.output = DeserializeJSON(executionResult.getOutput());  
  
}
```



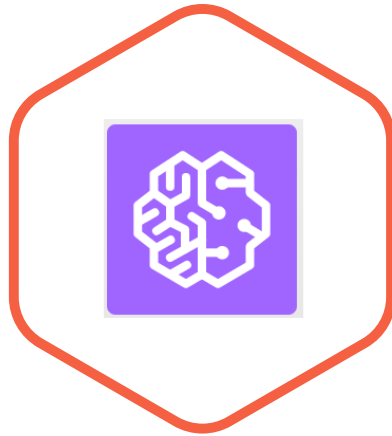
ASYNCHRONOUS STATUS CHECKING

- Scheduled task
- Persist execution ARNs

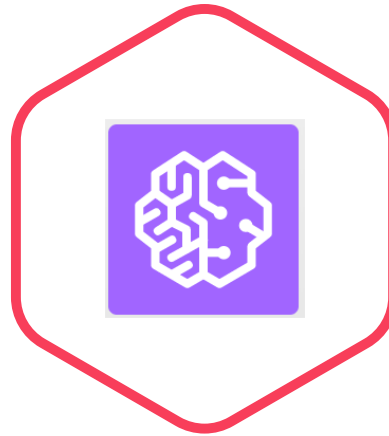
EXAMPLE WORKFLOW: TRANSCRIBE AND TRANSLATE A VIDEO



LAMBDA



TRANSCRIBE



TRANSLATE



POLLY



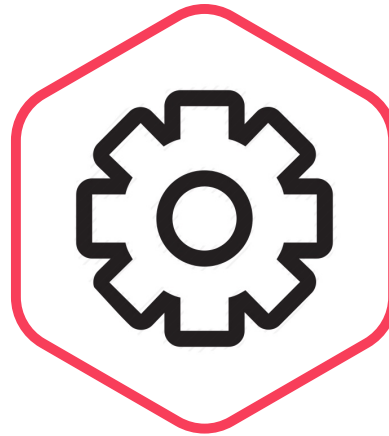
WHY IS THIS USEFUL?



ACCESSIBILITY



EFFICIENCY



UTILITY



INTERNATIONALIZATION



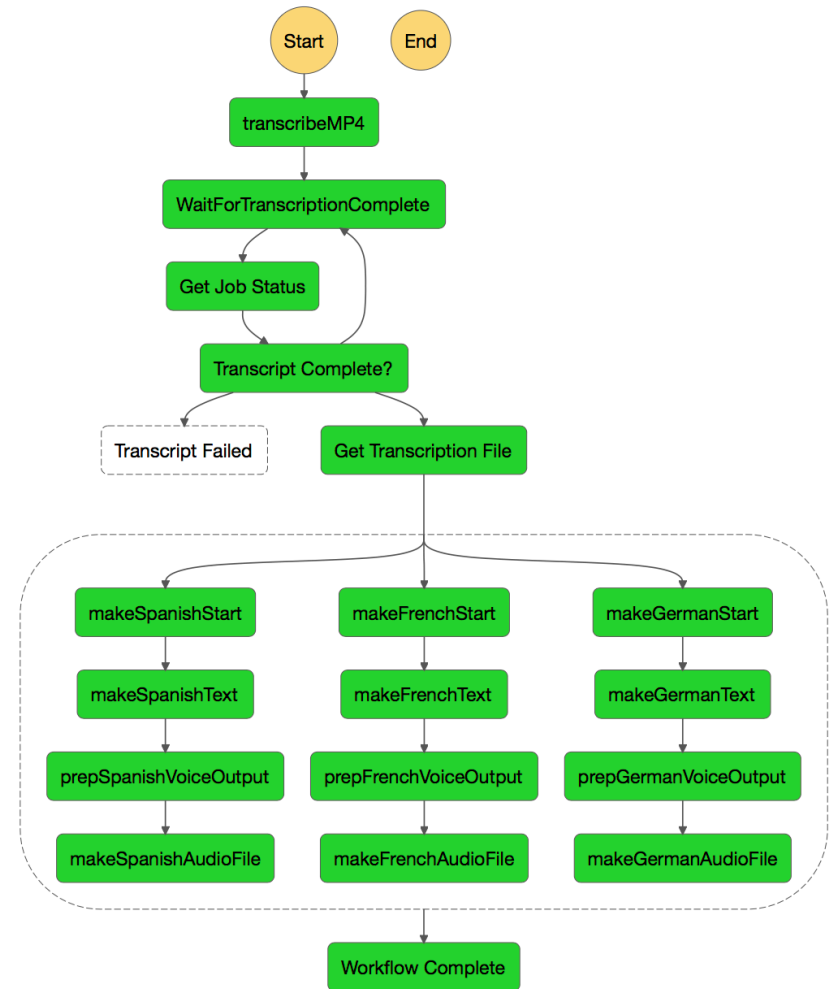
NEW STATE TYPES IN THE EXAMPLE

- ◆ Task with Retries
- ◆ Wait
- ◆ Parallel



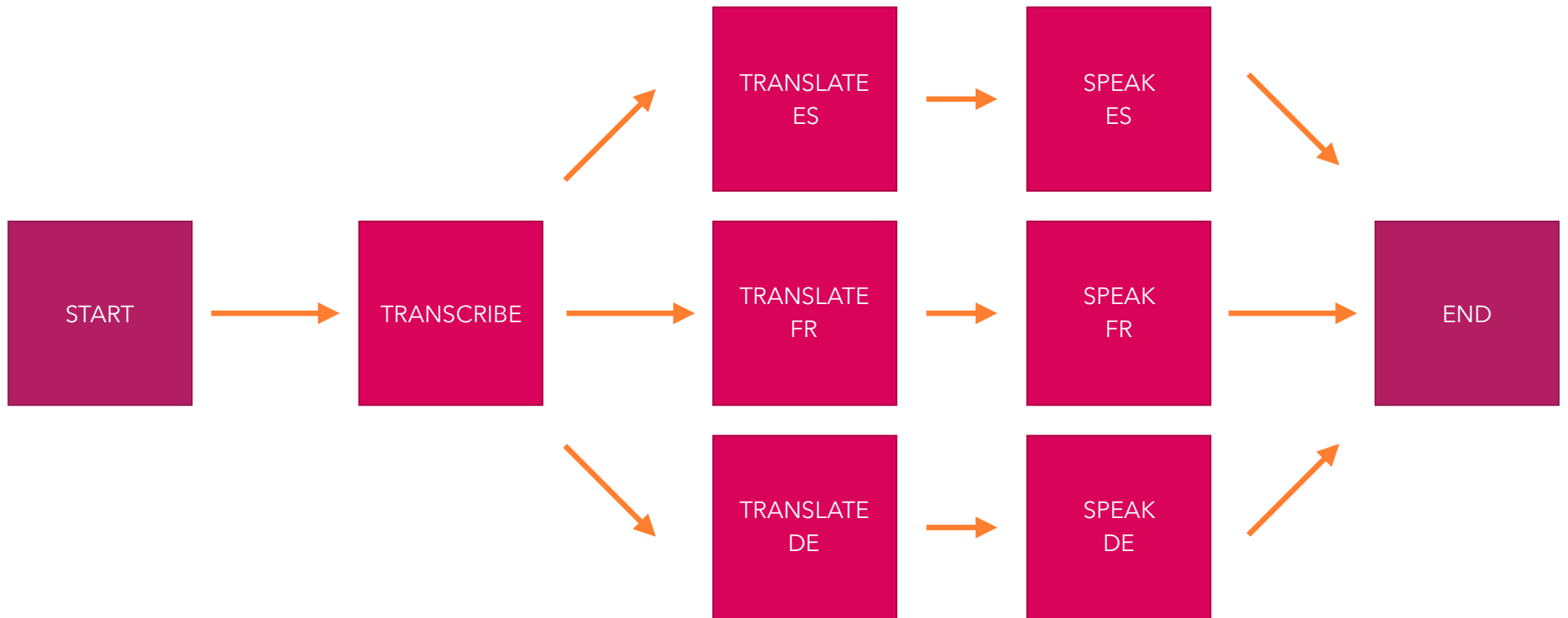
EXECUTION DIAGRAM

- Wait loop
- Parallel execution





WORKFLOW OVERVIEW





EXAMPLE 2: TRANSCRIBE, TRANSLATE, AND SPEAK



THROTTLING



EXAMPLE 2: TRANSCRIBE, TRANSLATE, AND SPEAK



**RETRIES ARE KEY
TO HANDLING
THROTTLED SERVICES**



KICKOFF TASK

```
"transcribeMP4": {  
  "Type": "Task",  
  "Resource": "arn:aws:lambda:us-east-1:XXXXXXXXXX:function:cfdemoStartTranscribeJob",  
  "Next": "WaitForTranscriptionComplete",  
  "Retry": [  
    {  
      "ErrorEquals": [ "States.ALL" ],  
      "IntervalSeconds": 30,  
      "MaxAttempts": 3,  
      "BackoffRate": 10  
    }  
  ]  
}
```




EXAMPLE 2: TRANSCRIBE, TRANSLATE, AND SPEAK



TRANSCRIBE



AWS TRANSCRIBE

- Accepts MP4, MP3, WAV, FLAC
- Understands English (US), Spanish (US)
- Transcriptions with punctuation
- Recognizes multiple speakers
- Timestamp each word
- Supports custom vocabulary for discipline-specific words
- \$0.0004 per second
- 60 minutes free per month



START TRANSCRIPTION JOB TASK

```
var returnData = {
  jobName: event.jobName
}

var params = {
  LanguageCode: 'en-US',
  Media: {
    MediaFileUri: event.srcFile
  },
  MediaFormat: event.mediaType,
  TranscriptionJobName: event.jobName
}
```

```
Transcribe.startTranscriptionJob(params, function(err,
data) {
  if (err) {
    console.log(err, err.stack);
    callback(err, null)
  } else {
    console.log(data); // successful response
    callback(null, returnData);
  }
});
```



EXAMPLE 2: TRANSCRIBE, TRANSLATE, AND SPEAK

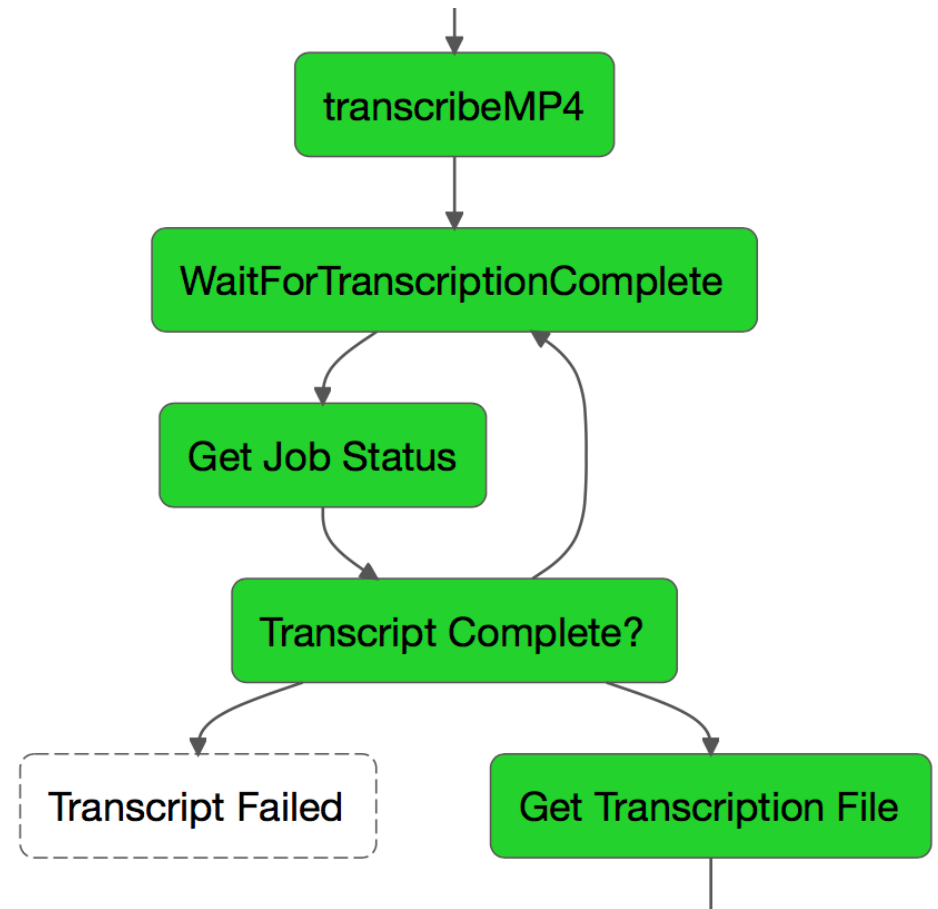
HOW DO WE HANDLE FULLY ASYNCHRONOUS TASKS?



CAPLINE HEADER ELEMENT

LOOPING IN A STEP FUNCTION

- Wait
- Check job status
- Job status done?





LOOPING IN STEP FUNCTIONS

```
"WaitForTranscriptionComplete": {
  "Type": "Wait",
  "Seconds": 30,
  "Next": "Get Job Status"
},
"Get Job Status": {
  "Type": "Task",
  "Resource": "arn:aws:lambda:us-
east-1:XXXXXXXX:function:cfdemoCheckTranscribeJobStatus",
  "Next": "Transcript Complete?"
},
```

```
"Transcript Complete?": {
  "Type": "Choice",
  "Choices": [
    {
      "Variable": "$.jobStatus",
      "StringEquals": "FAILED",
      "Next": "Transcript Failed"
    },
    {
      "Variable": "$.jobStatus",
      "StringEquals": "COMPLETED",
      "Next": "Get Transcription File"
    }
  ],
  "Default": "WaitForTranscriptionComplete"
}
```



WAIT STATES

- Seconds
- Timestamp
- Expiration date
- Timestamp path (variable)
 - "TimestampPath": "\$.expirationdate"



LOOPING IN STEP FUNCTIONS

```
"WaitForTranscriptionComplete": {  
  "Type": "Wait",  
  "Seconds": 30,  
  "Next": "Get Job Status"  
},
```

```
"Get Job Status": {  
  "Type": "Task",  
  "Resource": "arn:aws:lambda:us-  
east-1:XXXXXXXX:function:cfdemoCheckTranscribeJobStatus",  
  "Next": "Transcript Complete?"  
},
```

```
"Transcript Complete?": {  
  "Type": "Choice",  
  "Choices": [  
    {  
      "Variable": "$.jobStatus",  
      "StringEquals": "FAILED",  
      "Next": "Transcript Failed"  
    },  
    {  
      "Variable": "$.jobStatus",  
      "StringEquals": "COMPLETED",  
      "Next": "Get Transcription File"  
    }  
  ],  
  "Default": "WaitForTranscriptionComplete"  
}
```




CHECKING THE JOB STATUS

```
var params = { TranscriptionJobName: jobName }
var request = Transcribe.getTranscriptionJob(params, function(err, data) {
  if (err) { // an error occurred
    callback(err, null);
  } else { // successful response, return job status
    var returnData = {
      jobName: jobName,
      jobStatus: data.TranscriptionJob.TranscriptionJobStatus,
      transcriptFileUri: data.TranscriptionJob.Transcript.TranscriptFileUri,
      transcriptFileName: jobName
    };
    callback(null, returnData);
  }
});
```



LOOPING IN STEP FUNCTIONS

```
"WaitForTranscriptionComplete": {  
  "Type": "Wait",  
  "Seconds": 30,  
  "Next": "Get Job Status"  
},  
"Get Job Status": {  
  "Type": "Task",  
  "Resource": "arn:aws:lambda:us-  
east-1:XXXXXXXX:function:cfdemoCheckTranscribeJobStatus",  
  "Next": "Transcript Complete?"  
},
```

```
"Transcript Complete?": {  
  "Type": "Choice",  
  "Choices": [  
    {  
      "Variable": "$.jobStatus",  
      "StringEquals": "FAILED",  
      "Next": "Transcript Failed"  
    },  
    {  
      "Variable": "$.jobStatus",  
      "StringEquals": "COMPLETED",  
      "Next": "Get Transcription File"  
    }  
  ],  
  "Default": "WaitForTranscriptionComplete"  
}
```



GETTING THE TRANSCRIPTION RESULT

- Transcript does not go into a bucket you own!
- Time limited URI returned from `getTranscriptionJob()`

```
"TranscriptFileUri": "https://s3.amazonaws.com/aws-transcribe-us-east-1-prod/683830609677/confDemo-1523559305156/asrOutput.json?X-Amz-Security-Token=FQoDYXdzEEoaDFIZWqjw%2FgCc3kQGWSK3A43rTlruH70hezLPxIDyUEk%2Fk9Ga%2F6eRg7Hwpe5WpMoQQ0it..."
```

- Token is tied to account that generated the Transcribe job
- Grab and move to S3 for future use



SAVING THE TRANSCRIBE OUTPUT

```
getTranscript(transcriptFileUri).then(function(getTranscriptResponse) {
    return writeTranscriptToS3(getTranscriptResponse,transcriptFileName);
}).then(function(filePathOnS3) {
    var returnData = {
        transcriptFilePathOnS3: filePathOnS3,
        transcriptFileName: transcriptFileName
    };
    callback(null, returnData);
}).catch(function(err) {
    callback(err, null);
})
```



PROMISE ALL THE THINGS!

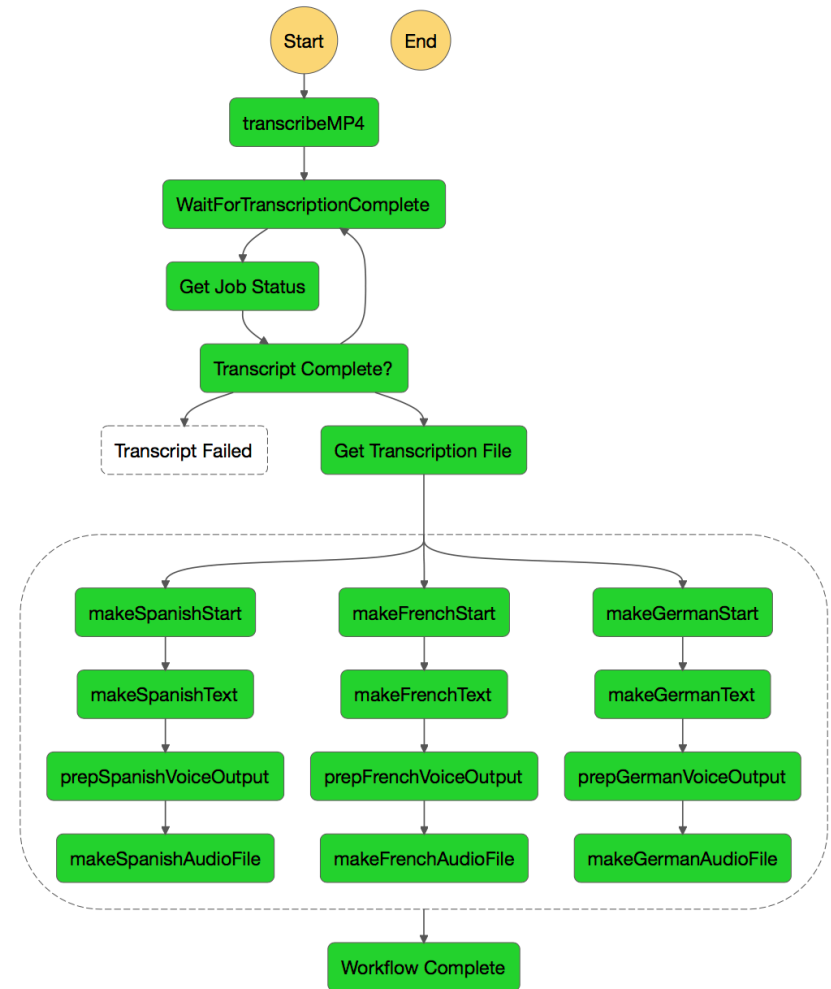
```
function getTranscript(transcriptFileUri) {
  return new Promise(function(resolve, reject) {
    https.get(transcriptFileUri, res => {
      res.setEncoding("utf8");
      let body = "";
      res.on("data", data => {
        body += data;
      });
      res.on("end", () => {
        body = JSON.parse(body);
        let transcript = body.results.transcripts[0].transcript;
        resolve(transcript);
      });
      res.on("error", (err) => {
        reject(Error(err));
      });
    });
  });
}
```

```
function writeTranscriptToS3(transcript, transcriptFileName) {
  return new Promise(function(resolve, reject) {
    let filePathOnS3 = 'transcripts/' + transcriptFileName + '.txt';
    var params = {
      Bucket: 'conferencedemobucket',
      Key: filePathOnS3,
      Body: transcript
    };
    var putObjectPromise = S3.putObject(params).promise();
    putObjectPromise.then(function(data) {
      resolve(filePathOnS3);
    }).catch(function(err) {
      reject(Error(err));
    });
  });
}
```



EXECUTION DIAGRAM

- Wait loop
- Parallel execution





EXAMPLE 2: TRANSCRIBE, TRANSLATE, AND SPEAK

PARALLEL TASKS

- Significantly speed workflow execution
- Cannot be dynamically generated



PARALLEL TASKS

```
"Make Versions": {
```

```
  "Type": "Parallel",
```

```
  "Next": "Workflow Complete",
```

```
  "Branches": [
```

```
{
  "StartAt": "makeSpanishStart",
  "States": {
    "makeSpanishStart": {},
    "makeSpanishText": {},
    "makeSpanishAudioFile": {}
  }, // end branch
```

Parallel
Branch 1

```
{
```

```
  "StartAt": "makeFrenchStart",
```

```
  "States": {
```

```
{
  "makeFrenchStart": {},
  "makeFrenchText": {},
  "makeFrenchAudioFile": {}
```

```
}, // end branch
```

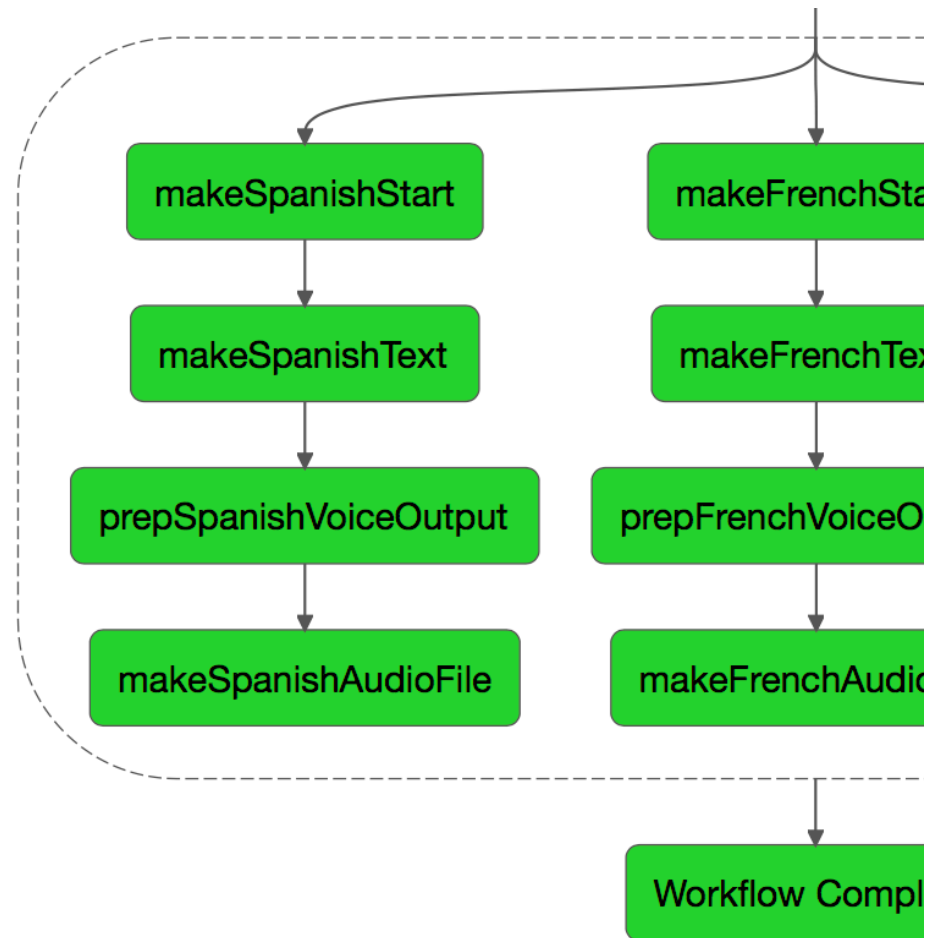
```
}}
```

Parallel
Branch 2



STEPS IN EACH PARALLEL TASK

- Define which language to use (Pass)
- Translate text (Task)
- Prep translated text for speech (Task)
- Speak translated text (Task)





DEFINE WHAT LANGUAGE TO USE

```
"makeSpanishStart": {  
  "Type": "Pass",  
  "Result": "es",  
  "ResultPath": "$.languageToUse",  
  "Next": "makeSpanishText"  
},
```



TRANSLATE ENGLISH TEXT

```
"makeSpanishText": {  
  "Type": "Task",  
  "Resource": "arn:aws:lambda:us-east-1:XXXXXXXX:function:cfdemoTranslateText",  
  "Next": "prepSpanishVoiceOutput",  
  "Retry": [  
    {  
      "ErrorEquals": [ "States.ALL" ],  
      "IntervalSeconds": 60,  
      "MaxAttempts": 3,  
      "BackoffRate": 5  
    }  
  ]  
}
```



EXAMPLE 2: TRANSCRIBE, TRANSLATE, AND SPEAK



TRANSLATE



AWS TRANSLATE

- To/from English only
 - Arabic (ar)
 - Chinese (Simplified) (zh)
 - French (fr)
 - German (de)
 - Portuguese (pt)
 - Spanish (es)
- Limit of 5000 characters per 10 seconds
- \$15 per million characters
- 2 million characters free per month



TRANSLATING TEXT

```
getTranscriptFile(transcriptFileOnS3).then(function(getTranscriptResponse) {
  return translateText(getTranscriptResponse, languageToUse);
}).then(function(translatedTextObj) {
  var returnData = {
    translatedText: translatedTextObj.TranslatedText,
    languageOfText: languageToUse,
    sourceTranscriptFileName: sourceTranscriptFileName
  }
  callback(null, returnData);
}).catch(function(err) {
  callback(err, null);
})
```



TRANSLATING TEXT

```
function translateText(textToTranslate, languageToUse) {
  return new Promise(function(resolve, reject) {
    // Current maximum length for translation of 5000 chars
    var maxLength = 4500;

    var trimmedString = textToTranslate.substr(0,
    maxLength);

    // We don't want to pass in words that are cut off
    trimmedString = trimmedString.substr(0,
    Math.min(trimmedString.length, trimmedString.lastIndexOf("
    ")));

    var params = {
      SourceLanguageCode: 'en',
      TargetLanguageCode: languageToUse,
```

```
      Text: trimmedString
    };

    Translate.translateText(params, function(err, data) {
      if (err) {
        reject(Error(err));
      } else {
        console.log("Successful translation:\n");
        resolve(data);
      }
    });
  });
}
```



TRANSLATE ENGLISH TEXT

```
"makeSpanishText": {  
  "Type": "Task",  
  "Resource": "arn:aws:lambda:us-east-1:XXXXXXXX:function:cfdemoTranslateText",  
  "Next": "prepSpanishVoiceOutput",  
  "Retry": [  
    {  
      "ErrorEquals": [ "States.ALL" ],  
      "IntervalSeconds": 60,  
      "MaxAttempts": 3,  
      "BackoffRate": 5  
    }  
  ]  
}
```




PREPARING THE TRANSLATED TEXT FOR SPEECH

```
"prepSpanishVoiceOutput": {  
  "Type": "Task",  
  "Resource": "arn:aws:lambda:us-east-1:XXX:function:cfDemoPrepTranslatedTextForSpeech",  
  "Next": "makeSpanishAudioFile"  
}
```



EXAMPLE 2: TRANSCRIBE, TRANSLATE, AND SPEAK

VARIABLE REASSIGNMENT IN STEP FUNCTIONS IS A PAIN



REASSIGNING VARIABLES

```
exports.handler = (event, context, callback) => {  
  var textToSpeak = event.translatedText;  
  var languageOfText = event.languageOfText;  
  var transcriptFileName = event.sourceTranscriptFileName;  
  
  var returnData = {  
    textToSpeak: textToSpeak,  
    languageOfText: languageOfText,  
    transcriptFileName: transcriptFileName  
  }  
  callback(null, returnData);  
};
```



TURNING TEXT INTO SPEECH

```
"makeSpanishAudioFile": {
  "Type": "Task",
  "Resource": "arn:aws:lambda:us-east-1:XXXXXXX:function:cfDemoConvertTextToSpeech",
  "Retry": [
    {
      "ErrorEquals": [ "States.ALL" ],
      "IntervalSeconds": 60,
      "MaxAttempts": 3,
      "BackoffRate": 5
    }
  ],
  "End": true
}
```



EXAMPLE 2: TRANSCRIBE, TRANSLATE, AND SPEAK



POLLY



AWS POLLY

- Male and female voices in
 - Danish
 - Dutch
 - English (AU, GB, GB-WLS, IN, US)
 - French (FR, FR-CA)
 - German
 - Icelandic
 - Italian
 - Japanese
 - Korean
 - Norwegian
 - Polish
 - Portuguese (BR, PT)
 - Romanian
 - Russian
 - Spanish (ES, US)
 - Swedish
 - Turkish
 - Welsh
- Limit of 100 requests per second
- \$4 per 1 million characters for speech
- 5 million characters free per month



TURNING TEXT INTO SPEECH

```
makeMP3FromText(textToSpeak, languageOfText).then(function(makeMP3Result) {  
    return writeFileToS3(makeMP3Result.AudioStream, languageOfText, fileNameForOutput);  
}).then(function(mp3FileNameOnS3) {  
    var returnData = {};  
    returnData["mp3FileNameOnS3-" + languageOfText] = mp3FileNameOnS3  
    callback(null, returnData);  
}).catch(function(err) {  
    callback(err, null);  
})
```



TURNING TEXT INTO SPEECH

```
function makeMP3FromText(textToSpeak, languageOfText) {
  return new Promise(function(resolve, reject) {
    var voiceToUse = 'lvy';
    // Polly has a current maximum character length of 3000 characters
    var maxLength = 2900;
    var trimmedText = textToSpeak.substr(0, maxLength);
    switch(languageOfText) {
      case 'es':
        voiceToUse = (Math.random() >= 0.5) ? "Penelope" : "Miguel"; break;
      case 'fr':
        voiceToUse = (Math.random() >= 0.5) ? "Celine" : "Mathieu"; break;
      case 'de':
        voiceToUse = (Math.random() >= 0.5) ? "Vicki" : "Hans"; break;
    }
  })
}
```




TURNING TEXT INTO SPEECH

```
var params = {
  OutputFormat: "mp3",
  SampleRate: "8000",
  Text: trimmedText,
  VoiceId: voiceToUse
}
var speakPromise = Polly.synthesizeSpeech(params).promise();
speakPromise.then(function(data) { // successfully created MP3 stream
  resolve(data);
}).catch(function(err) {
  reject(Error(err));
});
```



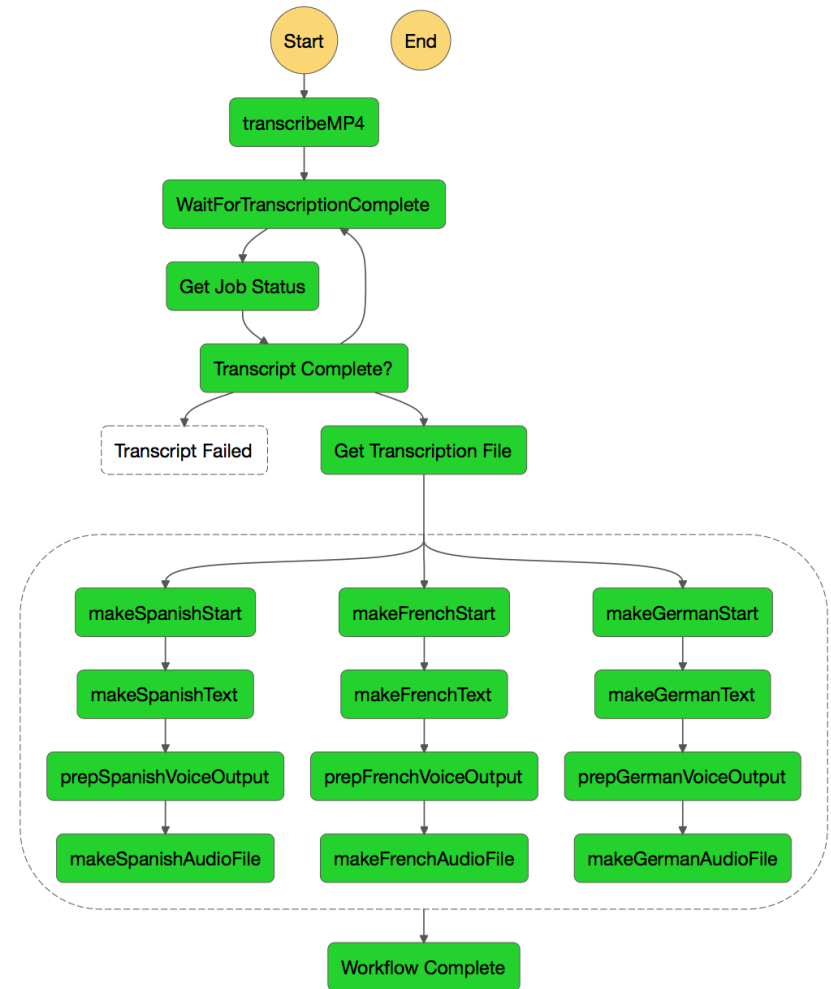
TURNING TEXT INTO SPEECH

```
"makeSpanishAudioFile": {  
  "Type": "Task",  
  "Resource": "arn:aws:lambda:us-east-1:XXXXXXX:function:cfDemoConvertTextToSpeech",  
  "Retry": [  
    {  
      "ErrorEquals": [ "States.ALL" ],  
      "IntervalSeconds": 60,  
      "MaxAttempts": 3,  
      "BackoffRate": 5  
    }  
  ],  
  "End": true  
}
```



EXECUTION DIAGRAM

- Wait loop
- Parallel execution





EXAMPLE 2: TRANSCRIBE, TRANSLATE, AND SPEAK

SO WHERE DOES **CF** COME IN?



CF IS THE SCHEDULER AND EXECUTOR



YOUR CFML NEEDS TO:

- Make an execution request
- Get the ARN of the execution for follow-up
- Check on completion
- If complete, get the results
- CF2018: Futures?



EXAMPLE 2: TRANSCRIBE, TRANSLATE, AND SPEAK

3 DAYS



PRODUCTION IMPROVEMENTS

- Improved handling of service limitations
- Logical chunking / chunk stitching
- Search video via Elasticsearch



WHAT'S NEXT?

GO DO!



WHAT'S NEXT?

AWS Playbox

<https://github.com/brianklaas/awsPlaybox>

brian.klaas@gmail.com

@brian_klaas