Data Engineer

3+ Week Onboarding Plan

Table Of Contents

Overview

Achievables

Mentor: determine exactly what the mentee knows and doesn't know, daily standup meeting to review tasks and ongoing support for questions / guidance.

Mentee: By the end of the 3rd week, they should know the basics.

Essential Services

By the end of the onboarding process, they should be familiar with these services/technologies:

AWS: S3, Lambda, Glue, Athena, Step Function, DynamoDB, RDS, DMS/SCT, Quicksight, all things VPC

Additional: Snowflake, Terraform

Week 1

Day 1

Goal: give a general overview of AllCloud and the Data Group, while gaining access to all relevant services for an employee.

Checklist:

- First day meetings (IT, finance, manager, etc)
- Access OneLogin, Confluence Data space, BitBucket, LucidChart
- Access Udemy: send a request
- Access Slack, Jira, Zoom
- Access personal AWS Sandbox Account
- Access data group AWS Sandbox Account
- Setup daily meeting with mentor
- Meeting with mentor: Bamboo, Jira, manager expectations, data group, company structure (engage, MSP, devops, etc), expectations for onboarding

Day 2

Goal: set up technical tools (PyCharm, DBeaver) and complete an S3 task.

Checklist:

- Set up budget alert on personal AWS Sandbox Account
- Download PyCharm with professional license and download Python 3
- Understand AWS programmatic profile (AWS Access ID and Secret Key): documentation
    - Download AWS CLI through terminal
- Connect PyCharm to AWS Sandbox account using IAM user: documentation
- Download DBeaver for DB client and Docker Desktop Client
- Task: S3 bucket replication task (steps detailed in last section of this document)
    - Set up rep rule in same account/ region: create two buckets (source, destination)
    - Set up rep rule in same account, different region: create new dest bucket in new region
    - Set up rep rule cross account, same region: create new dest bucket in data group account
- Meeting with mentor: Review task

Task to provide: In your sandbox AWS account, you will be creating replication rules between S3 buckets within the same region, between different regions and between different AWS accounts.

- Create a replication rule in the same account / same region
- Create a replication rule in the same account / different region
- Create a replication rule between your AWS sandbox account and the Data Group AWS account

Test each replication rule by uploading a file to the source bucket and checking the destination bucket.

Day 3

Goal: Dive in AWS cloud essentials

- Create AWS APN Account: https://partnercentral.awspartner.com/APNSelfRegister
- Complete AWS Technical Essentials Course

Day 4

Goal: Introduction to boto3 to complete first task of a simple pipeline

Checklist:

- Introduction to boto3: https://www.youtube.com/watch?v=SmilJDG4B_8
- Meeting with mentor: understand all things Lambda
    - Topics: navigate through Lambda service, triggers, general configuration, monitor through CW, configure test event, Lambda layers
- Create a Pandas Lambda Layer using confluence guide
- Task: create pipeline from S3 -> Lambda -> S3 (steps detailed in last section of this document)
    - Utilize previously created source and dest S3 bucket
    - Create Lambda function with AWS SDK Pandas Lambda Layer (SNS for error notification)
    - Draw up architecture
- Meeting with mentor: review task

Task to provide: In your sandbox AWS account, decouple the previously created source and destination S3 buckets with a Lambda function.

- Remove the previously created replication rule in the source bucket
- Create a Lambda function that uses the AWS SDK Lambda layer that processes the following input data file about data science salaries:
    - Function should: process the S3 create event to extract the bucket/object key details, add a column called salary_in_usd that converts all salaries to USD, split the data by work year to create 1 CSV file per year and write that file partitioned by year to the destination S3 bucket
    - https://aws-sdk-pandas.readthedocs.io/en/stable/tutorials/003%20-%20Amazon%20S3.html
- Create an SNS topic for failure notification in the Lambda function
- Debug the Lambda function using a test event and CloudWatch logs
- Draw up an architecture of the pipeline using Lucidchart

Day 5

Checklist:

- Complete items on backlog

Week 2

Day 1

Goal: Jump into hands on course with AWS Management Console

Checklist:

- Complete half of AWS Hands-On Udemy Course (sections 1, 2, 3, 4, 5, 7, 9)
- Meeting with mentor:
    - EC2: creation, use PEM keys to SSH vs use instance connect through console, ping the instance through telnet, create a snapshot

Day 2

Goal: Jump into hands on course with AWS Management Console

Checklist:

- Complete AWS Hands-On Udemy Course (sections 10, 11, 12, 13, 14)
- Meeting with mentor:
    - IAM: create roles vs policies, attach policies to roles, AWS managed vs custom policies, trust relationships for policies
    - CloudWatch: how to monitor AWS services through CW, CW Alarms
    - SNS: topic creation, subscription creation, various types of subscriptions

Day 3

Goal: Get hands-on with DMS/SCT

Checklist:

- Session on DMS/SCT: show customer use case (Skale)
- Download Microsoft Remote Desktop
- Documentation: https://docs.aws.amazon.com/dms/latest/userguide/Welcome.html
- Complete DMS Workshop: Introduction, Getting Started, Microsoft SQL Server to Amazon Aurora (PostgreSQL) and Microsoft SQL Server to Amazon S3
- Meeting with mentor: Review task

Day 4

Goal: Start into Data Engineer Udemy course to deep dive into S3, IAM, EC2

Checklist:

- Complete sections 1, 2, 3, 4, 5, 7, 8, 9 of Data Engineering using AWS Data Analytics Course
- Meeting with mentor

Day 5

Goal: Continue the Data Engineer Udemy course to deep dive into Lambda, Glue

Checklist:

- Complete sections 10, 11, 13, 14, 15, 17 of Data Engineering using AWS Data Analytics Course
- Meeting with mentor

Week 3

Day 1

Goal: Finish the Data Engineer Udemy course

Checklist:

- Complete sections 20, 21, 22, 23, 24 of Data Engineering using AWS Data Analytics Course
- DynamoDB Labs: https://amazon-dynamodb-labs.com/
    - Workshop: Hands on Lab for DynamoDB (Skip Explore DynamoDB CLI)
- Mentor meeting

Day 2

Goal: Complete workshops to prepare for the next task

Checklist:

- Athena Workshop: Introduction, Getting Started, Athena Basics, Athena Federation
- Glue Workshop: Intro, How to Start, Access Dataset, Lab 01 (skip Optional: Using AWS Cloudformation and Optional: Using Spark), Lab 05 (only ETL Job), Lab 06, Lab 08, Lab 10
- Mentor meeting

Day 3

Goal: Expand pipeline from W1 D5 to include SQS, Glue, Athena, Quicksight

Checklist:

- Task: Expand upon previous task (S3 to Lambda to S3) to include Glue, Athena, Quicksight (steps detailed in last section of this document)
    - Use CloudWatch logs for debugging
    - Glue Data Catalog: Create database, create classifier, create and run crawler
    - Athena: Set up query editor (create S3 bucket for results) and create views from tables
    - QuickSight: Set up data source to be Athena views using SPICE and create visualizations
    - Draw up architecture
- Meeting with mentor: Review task

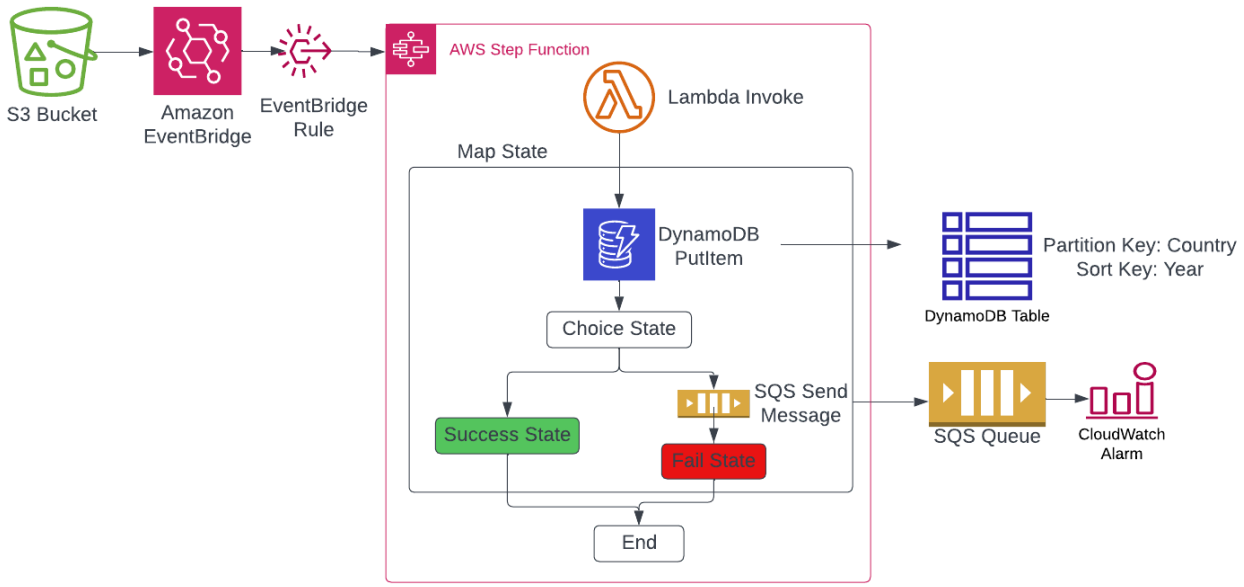
Task to provide: Building off of the previous task (S3 to Lambda to S3) by adding integration with Glue, Athena and Quicksight:

- Use: https://www.youtube.com/watch?v=V0GvZ_KAI70, https://blog.devgenius.io/creating-data-lake-using-aws-s3-glue-and-athena-9543849cb81d, https://aws.amazon.com/blogs/big-data/build-a-data-lake-foundation-with-aws-glue-and-amazon-s3/
- Create a table in the Glue Data Catalog that represents the data in the destination S3 bucket: need to create a database, a classifier and a crawler.
    - Want 1 table that represents the 3 different CSV files (partitioned by year) as one
- Use Athena to query the table (ensure that the year partition is visible) and create a view from that table. (https://www.w3schools.com/sql/sql_view.asp)
- Activate Quicksight, connect the data source to S3/Glue and create a dashboard with interesting visualizations of the data.
    - Use: https://docs.aws.amazon.com/quicksight/latest/user/welcome.html
- Expand upon the previously created architecture to include these components

Day 4 and beyond

Goal: Given an architecture and a description (similar to would be provided on an SOW), implement it manually in the console. Then, watch a Terraform course and write an IaaS application. Test the IaaS by launching it within the sandbox account with different names.

Task to provide: The architecture below displays a pipeline that a customer has requested.



A data file (called death_risk_factors.csv) will be inserted into an S3 bucket. This will send a notification to an EventBridge rule that will trigger the launch of a Step Function.

The Step Function will do the following:

1. Invoke a Lambda Function that will process the EventBridge rule event in order to: extract the bucket name, extract the object key, extract the data as a CSV file and iterate over each row of the CSV file to create a list of JSON objects, where each JSON object is:

{“country”: country_name, “year”: year, “risk_factor_1”: value1, “risk_factor_2”: value2, ...}

   The Lambda function returns this list of JSON objects.

2. Pass the output of the Lambda function to a Map State, which will iterate over each JSON object and do the following for each object:
   a. Use DynamoDB Put Item state to insert the JSON object to a DynamoDB table
   b. Use a Choice State to evaluate the HTTP Status Code returned from the response of the DynamoDB Put Item state:
      i. Return Success State if the HTTP Status Code is equal to 200
      ii. Return Fail State and use SQS Send Message to send a message to an SQS queue otherwise
         1. The SQS Queue should have a CloudWatch Alarm monitoring the number of items in the queue to send an email if it is > 1

Complete the following:

- Step 1: Review the architecture and description -> Questions?
- Step 2: Build the components manually in AWS Console -> Test end to end (2 days)
- Step 3: Learn Terraform -> 2 courses (no need to implement, but set up the environment)
    - https://learn.hashicorp.com/collections/terraform/aws-get-started
    - https://allcloud.udemy.com/course/terraform-fast-track/ (skip challenges)
- Step 4: Implement the manually built components into Terraform
    - Example documentation
    - https://bitbucket.org/emindsys/salt-security/src/master/
- Step 5: Present and demo for 20 minutes as if it was for a customer
    - Review given architecture and explain benefits of implementing as IaaS
    - Launch into sandbox with Terraform
    - Demo the arch by inserting input data (show SF running and DynamoDB table full)

Task Solutions

Task 1: S3 Bucket Replication

This task is a simple introduction to S3 by implementing three kinds of bucket replication:

Part1: Same account / same region bucket replication

- In sandbox AWS Account, create a source and destination bucket in us-east-1 region
    - Enable bucket versioning in both buckets
- In source bucket: go to Management tab -> Replication rules -> Create replication rule
    - Select all objects in the bucket
    - Choose bucket in this account -> destination bucket
    - IAM role -> Create new role
- Test by adding a file to source bucket

Part 2: Same account / different region bucket replication

- In sandbox AWS Account, create a new destination bucket in eu-west-1 region
    - Enable bucket versioning in new bucket
- In source bucket: go to Management tab -> Replication rules -> Create replication rule
    - Select all objects in the bucket
    - Choose bucket in this account -> destination bucket
    - IAM role -> Create new role
- Test by adding a file to source bucket

Part 3: Cross account / same region

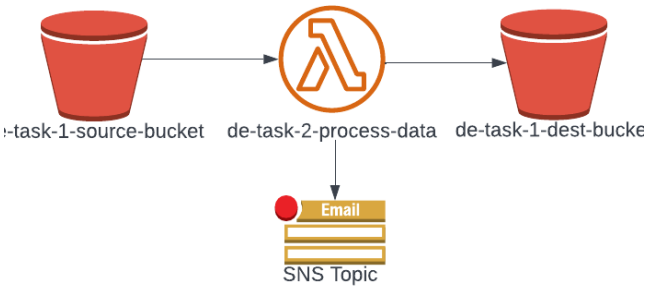
- In data group AWS Account, create a new destination bucket in us-east-1 region
    - Enable bucket versioning in new bucket
    - Copy AWS Account ID and bucket name
- In sandbox AWS account source bucket: go to Management tab -> Rep rules -> Create
    - Change object ownership to destination bucket owner
    - Copy AWS Account ID and bucket name
    - Copy IAM role associated with created replication rules
- In data group AWS Account, modify the role inside the guide and add it to the bucket policy of the new destination bucket
- Test by adding a file to source bucket

Task 2: S3 to Lambda to S3 with AWS SDK Layer

This task is to build a small pipeline to upload data to an S3 bucket, process it within Lambda using Pandas and output it to another S3 bucket.

- Delete any previous replication rules set up in source bucket

- In the sandbox AWS account, use the previously created source and dest buckets in us-east-1
    - In input S3 bucket, add trigger to the Lambda function
    - Input data: Data Science Salaries
- Create an IAM role for the Lambda Function and an IAM policy for the role:
    - IAM policy should have:
        - S3 list and read permissions for input S3 bucket
        - S3 list and write permissions for output S3 bucket
        - CloudWatch logs full permissions
        - SNS list and publish permissions for topics
    - Associate IAM policy with IAM role
- Create an SNS topic for failure notification with email receptor in the us-east-1 region
- Create a Lambda function and attach the new IAM role to it
    - Update Configuration to run for 10 minutes
    - Add an S3 create object test event to see how the event comes into the function
    - Function should: Process S3 event, extract bucket/object from event, complete basic data transformations using pandas and write output S3 bucket using boto3
        - Use AWS SDK for Pandas Guide
        - Add a column called salary_in_usd that converts all salaries to USD
        - Split data by work year CSV files in output bucket
            - {year}/df_{year}.csv
- Test with CloudWatch logs
- Draw up architecture on Lucidchart

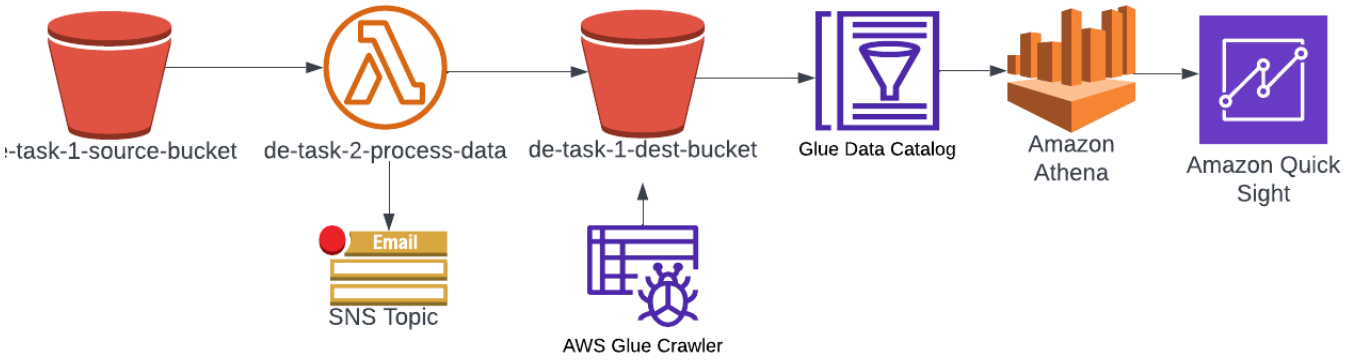


```python
import awswrangler as wr, boto3, json

def lambda_handler(event, context):
    conversion_rate_dict = { 'EUR': 1, 'USD' : 1, 'GBP' : 0.87, 'HUF' : 413.74, 'INR' : 82.41, 'JPY' : 147.39,
    'CNY' : 7.25, 'MXN' : 19.85,'CAD' : 1.36, 'DKK' : 7.47,'PLN' : 4.75, 'SGD' : 1.41, 'CLP' : 942.55}
    try:
        """======= Reading new CSV file that is entering to S3 source ======="""
        bucket = event['Records'][0]['s3']['bucket']['name']
        key = event["Records"][0]["s3"]["object"]["key"]
        df = wr.s3.read_csv([f"s3://{bucket}/{key}"])

        """ ======= Converting the workers' salaries to usd ======="""
        df['conversion_rate'] = df['salary_currency']
        df['conversion_rate'] = df['conversion_rate'].map(conversion_rate_dict)
        df['salary_in_usd'] = df.apply(lambda row: row['salary'] * row['conversion_rate'], axis=1)
        df.drop('conversion_rate', inplace=True, axis=1)

        """======= Spliting the input CSV file into multiple CSVs by years ======="""
        df.sort_values(by='work_year', axis=0, inplace=True)
        df.set_index(keys=['work_year'], drop=False,inplace=True)
        unique_years = df['work_year'].unique().tolist()
        for year in unique_years:
            x = df[df['work_year'] == year]
            wr.s3.to_csv(x, f"s3://allcloud-datateam-training-matan-dest/year={year}/{year}.csv", index=False)
    except:
        response = boto3.client('sns').publish(TopicArn='arn:aws:sns:eu-west-1:928737265640:salary-partition-task',
            Message=json.dumps({"S3Error" : 'Something went wrong!'}))
```

Task 3: S3 to Lambda to S3 to Glue to Athena to Quicksight

This task is to build upon the previous task to include Glue, Athena and Quicksight

- Lambda should do the same preprocessing and write data out to S3 output bucket
- Test the change from input S3 bucket to output S3 bucket -> use CW logs for debugging
- Create a database in the Glue data catalog
- Create a classifier that has headings and is in CSV format
- Create a crawler that uses the classifier and reads the CSV files from output data bucket
    - Crawler creation: create new IAM role in newly created database
- Run crawler and see the tables populated in Glue Data catalog
- In Athena, set up the query editor (create S3 bucket for Athena query results)
    - Create views based on tables
- In QuickSight, set up the data source to be Athena views using SPICE
    - Create basic visualizations and publish the dashboard
- Draw up the architecture in Lucidchart



Final Task: S3 to Step Function (Lambda, DynamoDB, SNS, SQS)

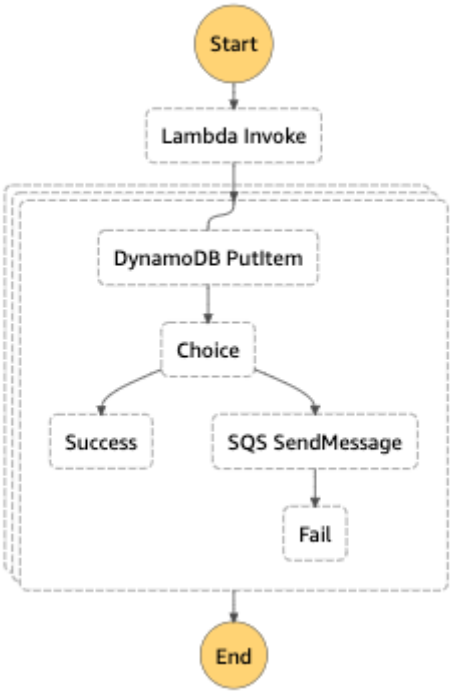
This task is to mimic the process of receiving a new project by receiving an architecture with description of the steps and sample data to ingest.

- Create a new S3 bucket to serve as the source.
    - In properties, turn on EventBridge notifications
- Create Step Function
- Create a rule in EventBridge:
    - Event source: AWS event
        - Event Pattern: Event source -> AWS Services, AWS Service -> S3, Event Type -> S3 Event Notification (Specific Event -> Object Created, Specific Bucket -> S3 bucket)
    - Target: AWS Service
        - Target -> Step Function state machine (select Step Function created)
- Test EventBridge rule by uploading the CSV file and seeing the execution in the SF
- Create a DynamoDB table with country as partition key and year as sort key (on-demand)
- Create an SQS queue
- Create a CloudWatch Alarm that checks if there is an item on the queue, then raise alarm and send an email
- Create a Lambda Function
    - IAM Policy should allow:
        - S3 GetObject for all objects in the bucket
- Modify the Step Function according to the architecture
    - IAM Policy should allow:
        - Lambda Invoke on created function
        - SQS Send Message on created queue
        - DynamoDB Put Item on create table
        - CloudWatch Logs full
- Develop the Lambda function:
    - Create a test event with EventBridge notification
    - Attach AWS SDK Lambda Layer

lambda_function × Execution results ×

```python
import json, sys
import awswrangler as wr

def lambda_handler(event, context):

    bucket_name = event['detail']['bucket']['name']
    object_name = event['detail']['object']['key']

    df = wr.s3.read_csv([f"s3://{bucket_name}/{object_name}"])

    items = []
    for index, row in df.iterrows():
        item = {
          "country": row['Entity'],
          "year": str(row['Year']),
          "outdoor_air_pollution": str(row["Outdoor Air Pollution"]),
          "drug_use": str(row["Drug Use"]),
          "low_birth_weight": str(row["Low Birth Weight"]),
          "unsafe_water": str(row["Unsafe Water"]),
          "unsafe_sanitation": str(row["Unsafe Sanitation"])
        }
        items.append(item)
    return items
```

- Develop the Step Function:


Start
Lambda Invoke
DynamoDB PutItem
Choice
Success
SQS SendMessage
Fail
End

```
{
  "Comment": "A description of my state machine",
  "StartAt": "Lambda Invoke",
  "States": {
    "Lambda Invoke": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "ResultPath": "$.items",
      "Parameters": {
        "Payload.$": "$",
        "FunctionName": "arn:aws:lambda:us-east-1:170091767549:function:de-final-task:$LATEST"
      },
      "Next": "Map"
    },
    "Map": {
      "Type": "Map",
      "InputPath": "$.items",
      "ItemsPath": "$.Payload",
      "Iterator": {
        "StartAt": "DynamoDB PutItem",
        "States": {
          "DynamoDB PutItem": {
            "Type": "Task",
            "Resource": "arn:aws:states:::dynamodb:putItem",
            "Parameters": {
              "TableName": "de-final-task",
              "Item": {
                "country": {
                  "S.$": "$.country"
                },
                "year": {
                  "N.$": "$.year"
                },
                "outdoor_air_pollution": {
                  "S.$": "$.outdoor_air_pollution"
                },
                "drug_use": {
                  "S.$": "$.drug_use"
                },
                "low_birth_weight": {
                  "S.$": "$.low_birth_weight"
                },
                "unsafe_water": {
                  "S.$": "$.unsafe_water"
                },
                "unsafe_sanitation": {
                  "S.$": "$.unsafe_sanitation"
                }
              }
            },
            "ResultPath": "$",
            "Next": "Choice"
          },
```


"unsafe_water": {
"S.$": "$.unsafe_water"
},
"unsafe_sanitation": {
"S.$": "$.unsafe_sanitation"
}
}
},
"ResultPath": "$",
"Next": "Choice"
},
"Choice": {
"Type": "Choice",
"Choices": [
{
"Variable": "$.SdkHttpMetadata.HttpStatusCode",
"NumericEquals": 200,
"Next": "Success"
},
{
"Not": {
"Variable": "$.SdkHttpMetadata.HttpStatusCode",
"NumericEquals": 200
},
"Next": "SQS SendMessage"
}
]
},
"Success": {
"Type": "Succeed"
},
"SQS SendMessage": {
"Type": "Task",
"Resource": "arn:aws:states:::sqs:sendMessage",
"Parameters": {
"MessageBody.$": "$",
"QueueUrl": "https://sqs.us-east-1.amazonaws.com/170091767549/de-final-task"
},
"Next": "Fail"
},
"Fail": {
"Type": "Fail"
}
}
},
"End": true
}
}
}