# CSYE7245 - Serverless Transactions Fraud Detection & Notification

**December  18, 2020**

Team-2

* Manish Singh
* Phu Tran
* Kaviya Sachi
* Rajavi Mehta

Overview

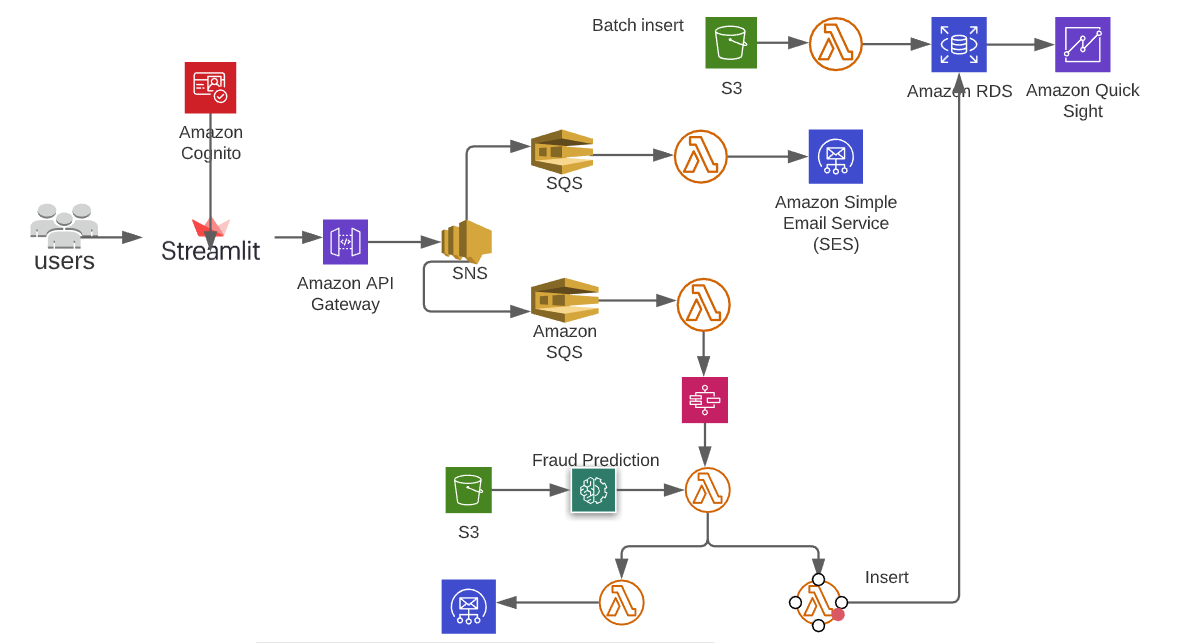
Did you know that each year, tens of billions of dollars are lost to online fraud world-wide?

Companies with online businesses have to constantly be on guard for fraudulent activity such as fake accounts and payments made with stolen credit cards.  One way they try to identify fraudsters is by using fraud detection apps, some of which use Machine Learning (ML).

Transaction Fraud Detection is a fully managed service that makes it easy to identify potentially fraudulent online Transaction such as online payment fraud.

Enter our Transaction Fraud Detection app. It uses historical data, ML to identify potentially fraudulent online activity so you can catch more fraud faster. You can create a fraud detection model with just a few clicks and no prior ML experience because Fraud Detector handles all of the ML heavy lifting for you.

Architecture



**Setup**

The pipeline requires an Amazon Web Services account to deploy and run. Signup for an AWS Account here. The pipeline uses the folllowing AWS Services:

* Amazon cognito
* Api Gateway
* Amazon SNS
* Amazon SQS
* Amazon SES
* Step Function
* Lambda Function
* Amazon Fraud Detector
* Amazon S3
* Amazon RDS
* Amazon Quick Sight

# Deployment and Execution

## Prerequisites

1. Download and install the latest version of Python for your OS from [here](https://www.python.org/downloads/). We shall be using Python 3.6 and above.
2. You will be needing [AWS CLI](http://docs.aws.amazon.com/cli/latest/userguide/installing.html) as well. If you already have AWS CLI, please upgrade to a minimum version of 1.16.67.
   1. pip install awscli --upgrade --user

Note: If you are using the Windows version of AWS CLI, please use the [Windows msi](https://docs.aws.amazon.com/cli/latest/userguide/install-windows.html#install-msi-on-windows) to upgrade your installation.

## Instructions

This code depends on a bunch of libraries (not included in this distribution) which you will have to install yourself. The code comes with a SAM template, which you can use to deploy the entire solution.

1. Download the contents of this repository on your local machine (say: project-directory)
2. The solution is implemented in python, so make sure you have a working python environment on your local machine.
3. Open a command prompt, navigate to the project directory. Navigate to the /code/lib sub directory and install the following libraries:
   1. pip install pymysql --target .
4. Create a S3 bucket for deployment (note: use the same region throughout the following steps, I have used us-west-2, you can replace it with the region of your choice. Refer to the [region table](https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services/) for service availability.)
   1. aws s3 mb s3://rds-lambda-demo-2018-us-west-2 --region us-west-2
5. Navigate to the /code/source sub directory. Package the contents and prepare deployment package using the following command
   1. aws cloudformation package --template-file LambdaRDS\_Demo.yaml --output-template-file LambdaRDS\_Demo\_output.yaml --s3-bucket rds-lambda-demo-2018-us-west-2 --region us-west-2
6. If you want to make changes to the Lambda functions, you can do so on your local machine and redeploy them using the steps 5 through 7 above. The package and deploy commands take care of zipping up the new Lambda files (along with the dependencies) and uploading them to AWS for execution.

## **Outputs**

Following are the outputs getting generated

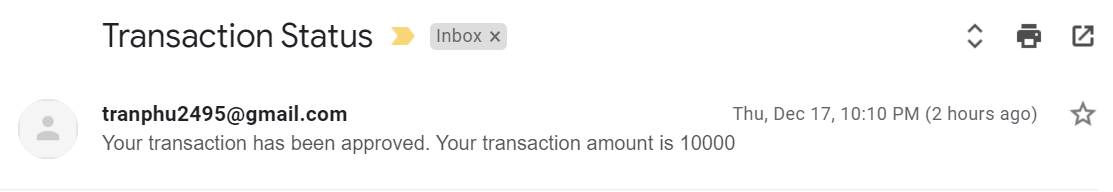
1. **RDS MySQL Server:** Insert of records to RDS after the prediction has been made
2. **Email :** An email notification to the customer, about their transaction- whether it is getting processed or not
3. **Quicksight Insight:** Insight of various customer transactions
4. **Cognito:** token generation to allow only valid users

## Execution

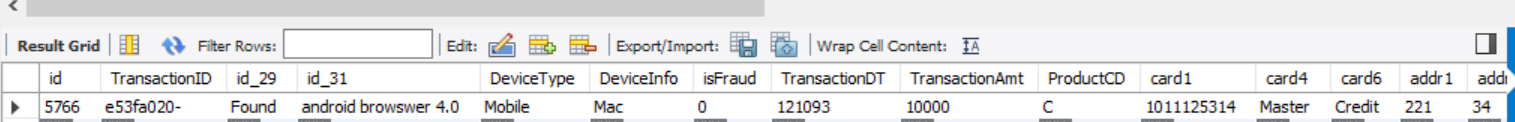
The csv file contains sample transaction records from [IEEE CIS Fraud Detection](https://www.kaggle.com/c/ieee-fraud-detection) that you can use to trigger the insert into rds and S3 to train the model. Navigate to your Lambda console, look for the function **LambdaProxy** and create a test event. You can modify the test iterations count to your choice.

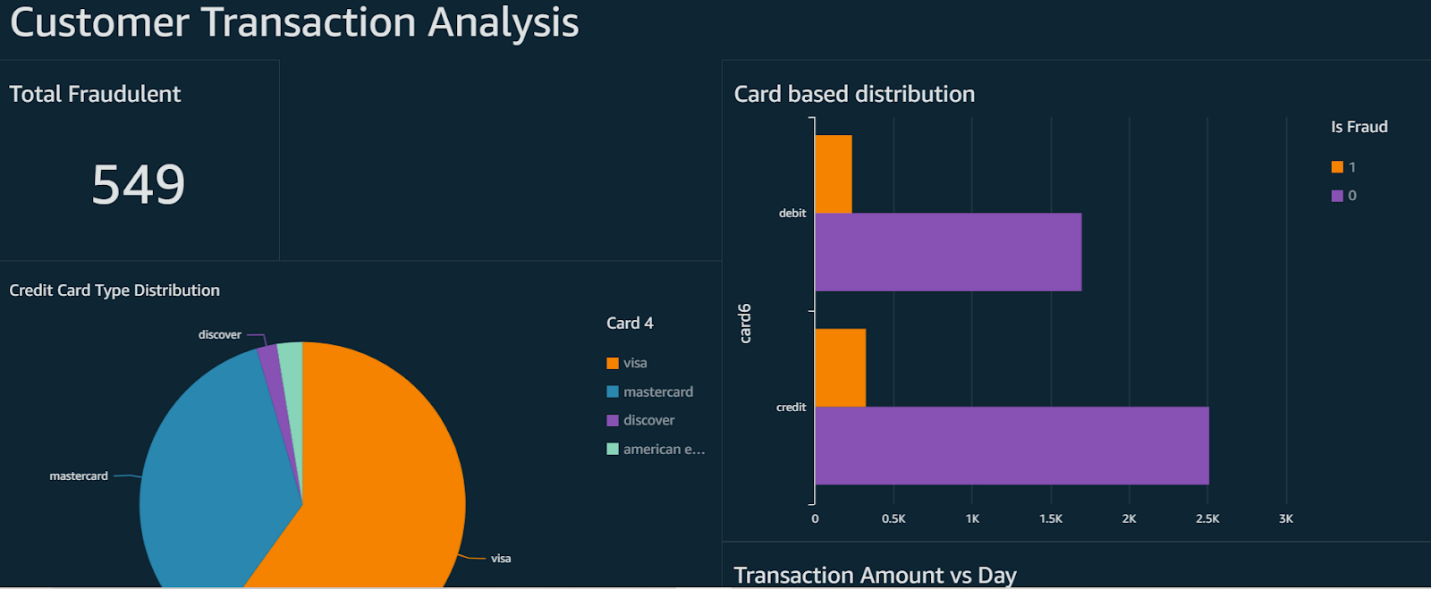
Once the test finishes, you can navigate to your CloudWatch console and use Customer Analytics Dashboard to look at the metrics. Also check your RDS database to see for the record being inserted and check your email for the notification

**Email Notification**



**RDS Table**

****

****

# Code Walkthrough

1. **Lambda\_s3tords\_bulkload**: Lambda function to simulate creating and inserting records from S3 to RDS
2. **Lambda\_transactionconfimedemail**: Lambda function to simulate sending a transaction confirmed email
3. **LambdaProxy**: helps to connect with Amazon Email service
4. **Step\_insert\_rds**: Lambda function helps to insert records into rds after prediction has been made
5. **Step\_sendemail**: Lambda function helps to send email message, based on the outcome of prediction
6. **Step\_transaction\_fraud\_detector**: retrieves outcome(low risk/highrisk) from fraud detector and pass it as input to step functions