Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Programme: MCA

Semester: Winter 24-25

Course: Cloud Computing

Course Code: PMCA506L

Faculty: Dr. Priyaadharshini

Slot: E2 + TE2

REAL-TIME CREDIT CARD FRAUD DETECTION USING AWS SERVICES

Team Members

24MCA1018 – Sivas D

24MCA1031 – Gokulnath E

24MCA1033 – Shajith N

Abstract

Credit card fraud is a significant financial issue that results in billions of dollars in losses each year. The need for an efficient, real-time fraud detection system is crucial for financial institutions. This project leverages AWS cloud services and machine learning to build an automated fraud detection system. The system classifies transactions as either fraudulent or legitimate based on various transaction features. Using AWS Lambda, API Gateway, SageMaker, SNS, CloudWatch, IAM, and EC2, we ensure a scalable and efficient solution. The implementation of this project highlights the real-time detection capabilities, alert notifications, and cloud-based security features that improve fraud prevention strategies

1. Introduction

1.1 Project Overview

Credit card fraud is a growing threat, leading to financial losses and security concerns. This project focuses on utilizing **AWS cloud technologies** to implement a **machine learning-powered fraud detection system** that can process transactions in real-time and provide instant alerts for suspected fraud cases.

1.2 Objectives

- Develop an automated fraud detection system with real-time classification.
- Utilize machine learning models to analyze transaction patterns.
- Deploy the system using AWS Lambda, API Gateway, SageMaker, SNS, CloudWatch, IAM, EC2, and S3.
- Ensure high accuracy in fraud detection.
- Provide instant alerts to users in case of suspected fraud.

Detailed Description of the Modules:

1. Amazon S3 (Simple Storage Service)

Purpose:
Used for storing datasets used to train the fraud detection model and storing preprocessed transaction data.

Functions:

- Uploads the raw credit card transaction dataset.
- Lambda fetches and preprocesses data from S3 before model training.
- Ensures durability and easy accessibility for SageMaker.

Interaction:
Lambda and SageMaker access data from S3 for training or inference.

2. Amazon SageMaker

Purpose:
To train and host the machine learning model for fraud detection.

Functions:

- Trains an XGBoost model using historical transaction data.
- Deploys a real-time endpoint for fraud prediction.
- Accepts feature inputs in CSV format and returns a fraud score.

Interaction:
Lambda invokes the SageMaker endpoint to get predictions during runtime.

3. AWS Lambda

Purpose:
Acts as the **serverless backend function** that handles business logic.

Functions:

- Parses API request JSON to extract features.
- Sends features to SageMaker for inference.
- Processes model output to determine if it's fraud.
- Sends alerts to SNS if fraud is detected.
- Logs to CloudWatch for traceability.

Interaction:
Triggered by API Gateway. Communicates with SageMaker, SNS, and CloudWatch.

4. Amazon API Gateway

Purpose:
Acts as the **front-facing interface** for users and clients to interact with the backend logic.

Functions:

- Accepts HTTP requests from the frontend or external tools (like `curl` or Postman).
- Forwards the request to Lambda securely.
- Handles CORS and throttling policies.

Interaction:
Receives input from EC2 (Flask UI) or command line, and routes to Lambda.

5. AWS SNS (Simple Notification Service)

Purpose:
Provides **real-time alerting** in the event of a fraud detection.

Functions:

- Sends email or SMS alerts when fraud is detected.
- Helps in rapid incident response or further transaction investigation.

Interaction:
Called by Lambda if the fraud score crosses a defined threshold.

6. AWS CloudWatch

Purpose:
Used for **monitoring and logging** activity across AWS services.

Functions:

- Logs Lambda executions and errors.
- Tracks performance metrics for debugging.
- Can trigger alarms or auto-remediation based on rules.

Interaction:
Lambda automatically pushes logs. IAM policies enable CloudWatch access.

7. AWS IAM (Identity and Access Management)

Purpose:
Provides **secure access control** to AWS services and resources.

Functions:

- Grants permissions to Lambda to invoke SageMaker, access SNS, S3, etc.
- Ensures only authorized actions are allowed.
- Manages roles (like InvokeSageMakerLambdaRole) with least-privilege policy.

Interaction:
Every service interaction is validated by IAM permissions behind the scenes.

8. Amazon EC2 (Elastic Compute Cloud)

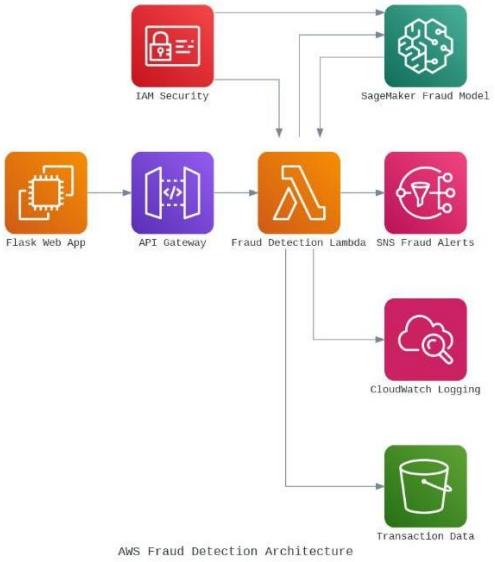
Purpose:
Hosts a **Flask web interface** for user input and model interaction.

Functions:

- Web server for transaction input form.
- Communicates with API Gateway.
- Uses Gunicorn and Nginx for production-grade serving.

Interaction:
Users access this UI to simulate transactions, which are then routed to the API.

2. Architecture & Workflow



AWS Fraud Detection Architecture

2.1 Workflow Steps

1. User submits a transaction request via a Flask UI hosted on EC2.
2. The request is forwarded to API Gateway.
3. API Gateway triggers AWS Lambda, which processes the transaction data.
4. Lambda invokes the SageMaker ML model to classify the transaction.
5. If fraud is detected, AWS SNS sends an alert notification.
6. Transaction records and logs are stored in S3 and CloudWatch for monitoring.

Individual Descriptions:

24MCA1033 – Shajith N

SageMaker, API Gateway, and Lambda:

- Developed and trained the XGBoost model using Amazon SageMaker on the credit card fraud dataset.

- Created and deployed a SageMaker endpoint to enable real-time inference from transaction features.

- Built AWS Lambda functions that:
    - Process input data.
    - Invoke the SageMaker endpoint.
    - Interpret the model prediction.
- Configured Amazon API Gateway to expose REST endpoints for transaction inputs.
- Ensured integration between API Gateway and Lambda with proper method and stage deployments.

24MCA1031 – Gokulnath E

EC2, SNS:

- Configured the EC2 instance (Ubuntu) with Python, Flask, Gunicorn, and Nginx.
- Developed the Flask-based UI to allow users to input transaction data easily.
- Integrated the frontend with the API Gateway endpoint for real-time requests.
- Set up AWS SNS to send fraud alerts via email/SMS from Lambda.
- Tested the end-to-end communication from UI to backend fraud detection system.

24MCA1018 – Sivas D

S3, IAM, CloudWatch:

Handled Amazon S3 storage:

- Uploaded datasets.
- Enabled access for training and Lambda pre-processing.

Created and managed IAM roles and policies:

- Defined secure permissions for Lambda, SageMaker, and API Gateway.
- Ensured least privilege access for all services.

Set up CloudWatch monitoring:

- Verified logs for Lambda executions and SNS triggers.
- Enabled performance and error monitoring.

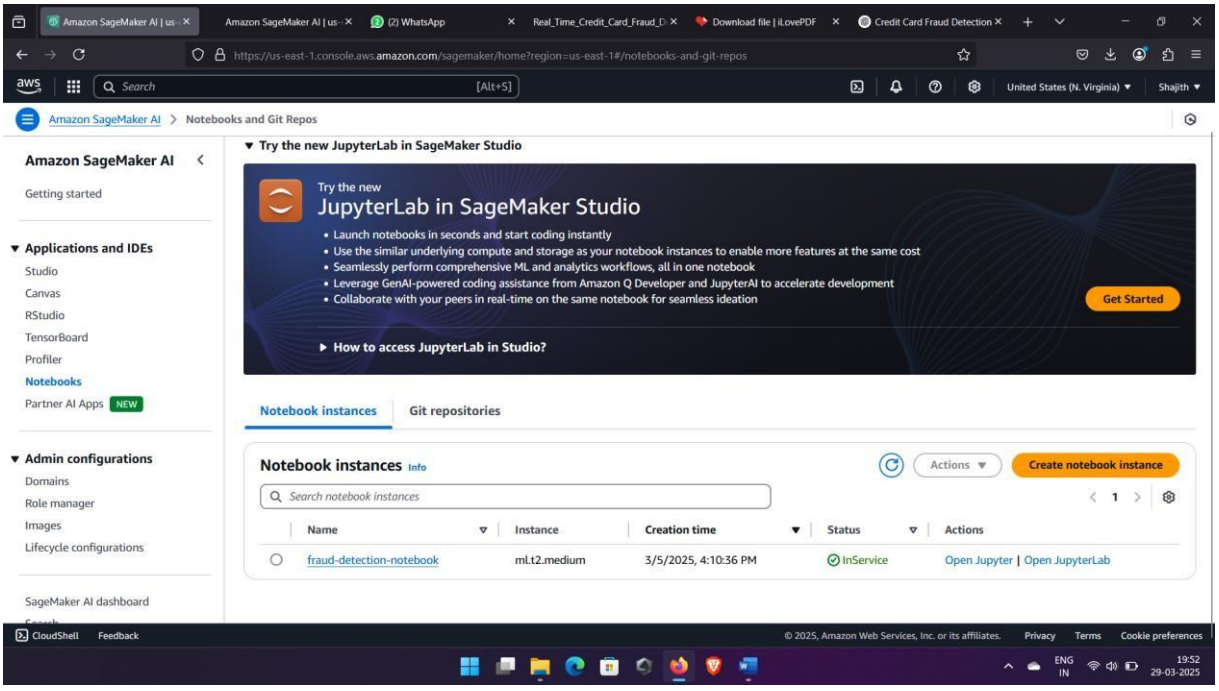
3. Implementation

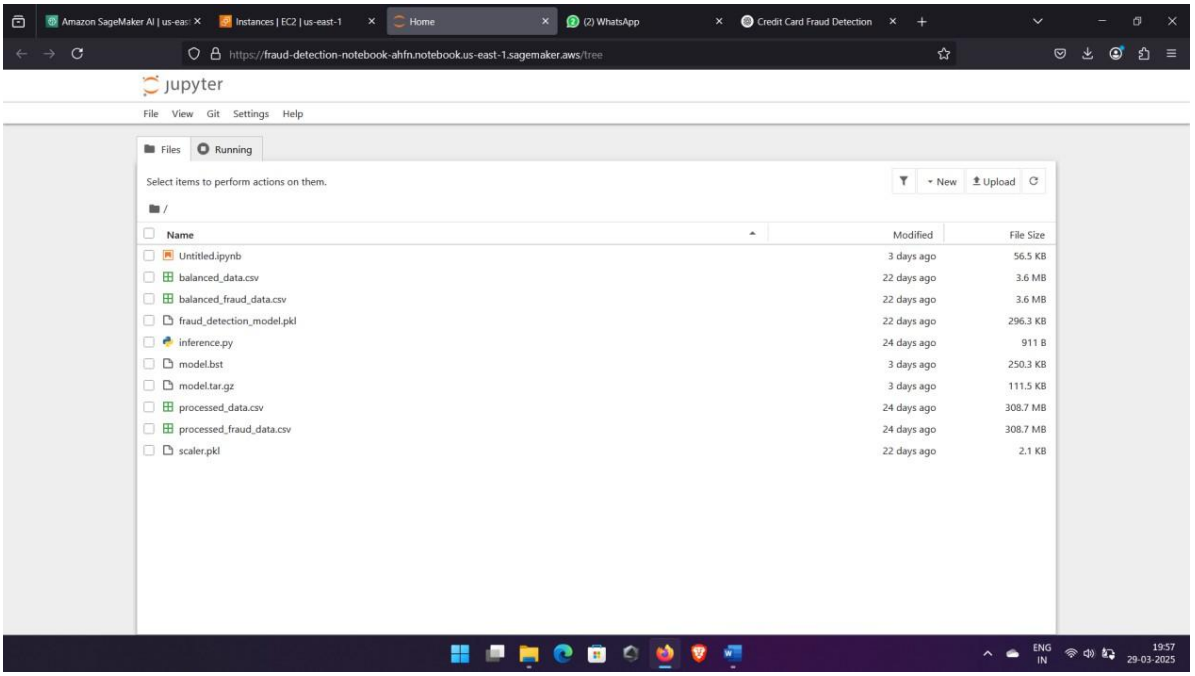
AWS Services Used

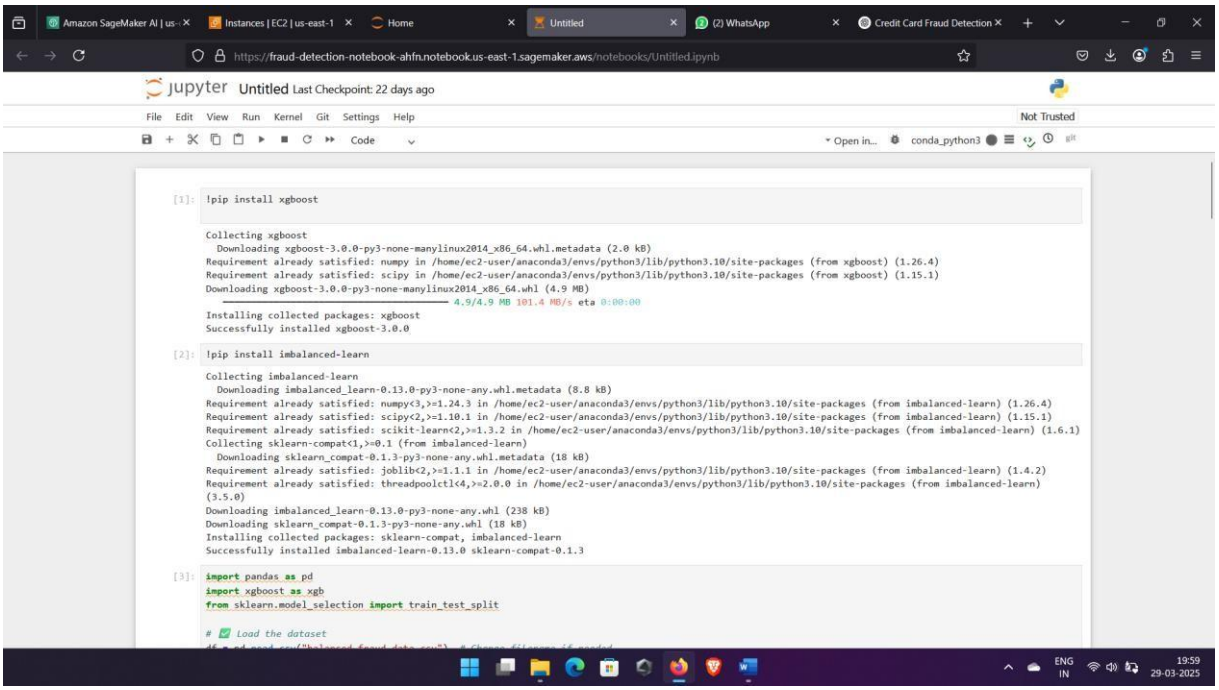
AWS SageMaker

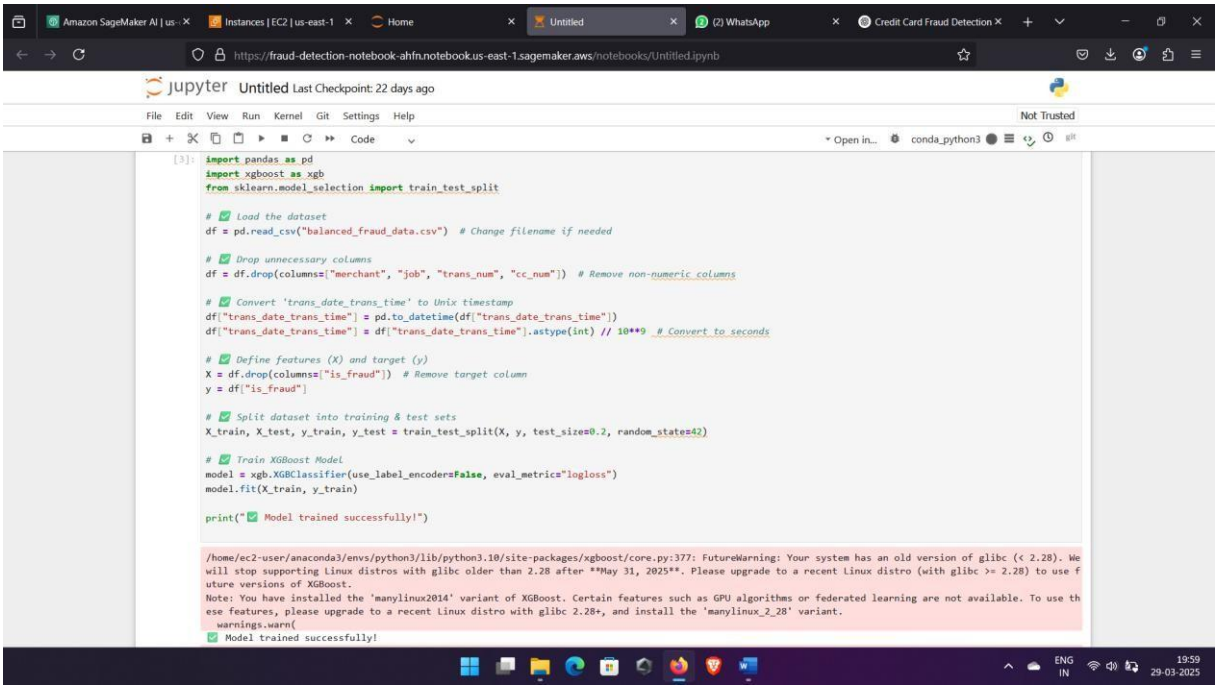
- Trained an XGBoost model on a credit card fraud dataset.
- Model predicts a fraud score, classifying transactions as fraudulent or non-fraud

Screenshot:

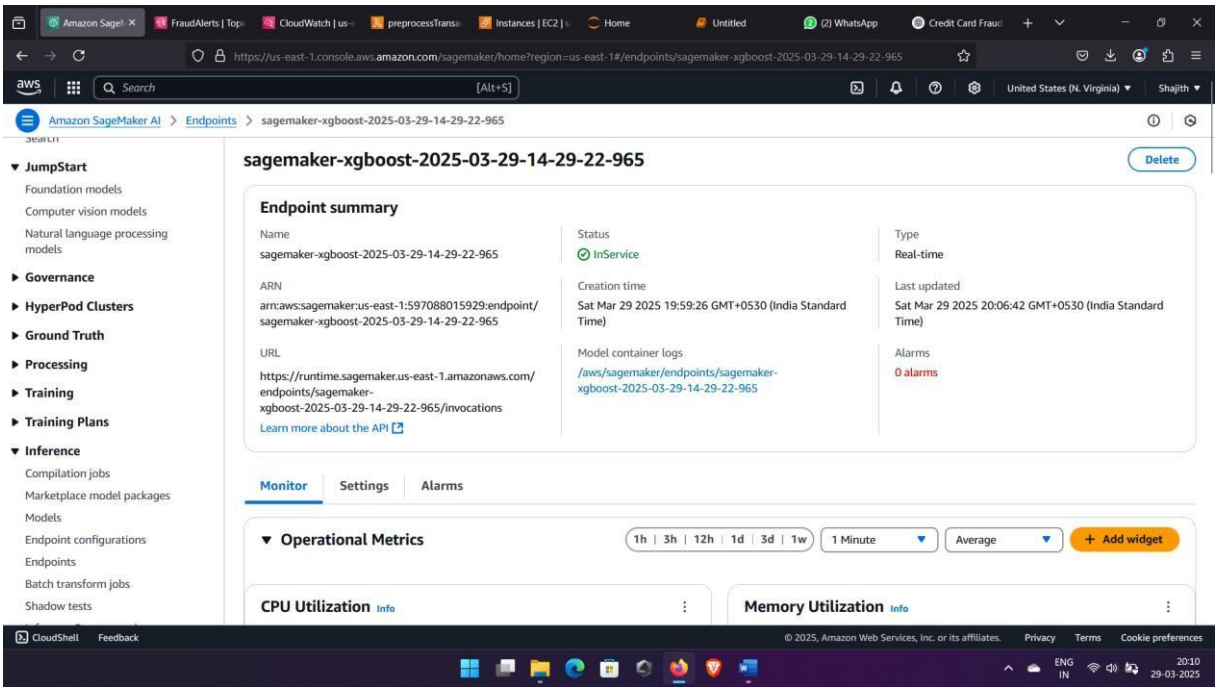



Amazon SageMaker AI | us-eas
Instances | EC2 | us-east-1
Home
(2) WhatsApp
Credit Card Fraud Detection
https://fraud-detection-notebook-ahfn.notebook.us-east-1.sagemaker.aws/tree
jupyter
File View Git Settings Help
Files
Running
Select items to perform actions on them.
New
Upload
/
Name
Modified
File Size
Untitled.ipynb 3 days ago 56.5 KB
balanced_data.csv 22 days ago 3.6 MB
balanced_fraud_data.csv 22 days ago 3.6 MB
fraud_detection_model.pkl 22 days ago 296.3 KB
inference.py 24 days ago 911 B
model.bst 3 days ago 250.3 KB
model.tar.gz 3 days ago 111.5 KB
processed_data.csv 24 days ago 308.7 MB
processed_fraud_data.csv 24 days ago 308.7 MB
scaler.pkl 22 days ago 2.1 KB
ENG
IN
19:57
29-03-2025


Amazon SageMaker AI | us-
Instances | EC2 | us-east-1
Home
Untitled
(2) WhatsApp
Credit Card Fraud Detection
https://fraud-detection-notebook-ahfn.notebook.us-east-1.sagemaker.aws/notebooks/Untitled.ipynb
jupyter Untitled Last Checkpoint: 22 days ago
File Edit View Run Kernel Git Settings Help
Not Trusted
Code
Open in...
conda_python3
[1]: !pip install xgboost
Collecting xgboost
Downloading xgboost-3.0.0-py3-none-manylinux2014_x86_64.whl.metadata (2.0 kB)
Requirement already satisfied: numpy in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from xgboost) (1.26.4)
Requirement already satisfied: scipy in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from xgboost) (1.15.1)
Downloading xgboost-3.0.0-py3-none-manylinux2014_x86_64.whl (4.9 MB)
4.9/4.9 MB 101.4 MB/s eta 0:00:00
Installing collected packages: xgboost
Successfully installed xgboost-3.0.0
[2]: !pip install imbalanced-learn
Collecting imbalanced-learn
Downloading imbalanced_learn-0.13.0-py3-none-any.whl.metadata (8.8 kB)
Requirement already satisfied: numpy<3,>=1.24.3 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from imbalanced-learn) (1.26.4)
Requirement already satisfied: scipy<2,>=1.10.1 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from imbalanced-learn) (1.15.1)
Requirement already satisfied: scikit-learn<2,>=1.3.2 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from imbalanced-learn) (1.6.1)
Collecting sklearn-compat<1,>=0.1 (from imbalanced-learn)
Downloading sklearn_compat-0.1.3-py3-none-any.whl.metadata (18 kB)
Requirement already satisfied: joblib<2,>=1.1.1 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from imbalanced-learn) (1.4.2)
Requirement already satisfied: threadpoolctl<4,>=2.0.0 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from imbalanced-learn)
(3.5.0)
Downloading imbalanced_learn-0.13.0-py3-none-any.whl (238 kB)
Downloading sklearn_compat-0.1.3-py3-none-any.whl (18 kB)
Installing collected packages: sklearn-compat, imbalanced-learn
Successfully installed imbalanced-learn-0.13.0 sklearn-compat-0.1.3
[3]: import pandas as pd
import xgboost as xgb
from sklearn.model_selection import train_test_split
# ✅ Load the dataset
ENG IN
19:59 29-03-2025


Amazon SageMaker AI | us-
Instances | EC2 | us-east-1
Home
Untitled
(2) WhatsApp
Credit Card Fraud Detection
https://fraud-detection-notebook-ahfn.notebook.us-east-1.sagemaker.aws/notebooks/Untitled.ipynb
jupyter Untitled Last Checkpoint: 22 days ago
File Edit View Run Kernel Git Settings Help
Not Trusted
Code
Open in...
conda_python3
[3]: import pandas as pd
import xgboost as xgb
from sklearn.model_selection import train_test_split
# ✅ Load the dataset
df = pd.read_csv("balanced_fraud_data.csv") # Change filename if needed
# ✅ Drop unnecessary columns
df = df.drop(columns=["merchant", "job", "trans_num", "cc_num"]) # Remove non-numeric columns
# ✅ Convert 'trans_date_trans_time' to Unix timestamp
df["trans_date_trans_time"] = pd.to_datetime(df["trans_date_trans_time"])
df["trans_date_trans_time"] = df["trans_date_trans_time"].astype(int) // 10**9 # Convert to seconds
# ✅ Define features (X) and target (y)
X = df.drop(columns=["is_fraud"]) # Remove target column
y = df["is_fraud"]
# ✅ Split dataset into training & test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# ✅ Train XGBoost Model
model = xgb.XGBClassifier(use_label_encoder=False, eval_metric="logloss")
model.fit(X_train, y_train)
print("✅ Model trained successfully!")
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/xgboost/core.py:377: FutureWarning: Your system has an old version of glibc (< 2.28). We
will stop supporting Linux distros with glibc older than 2.28 after **May 31, 2025**. Please upgrade to a recent Linux distro (with glibc >= 2.28) to use f
uture versions of XGBoost.
Note: You have installed the 'manylinux2014' variant of XGBoost. Certain features such as GPU algorithms or federated learning are not available. To use th
ese features, please upgrade to a recent Linux distro with glibc 2.28+, and install the 'manylinux_2_28' variant.
warnings.warn(
✅ Model trained successfully!
ENG IN
19:59 29-03-2025

Endpoint:

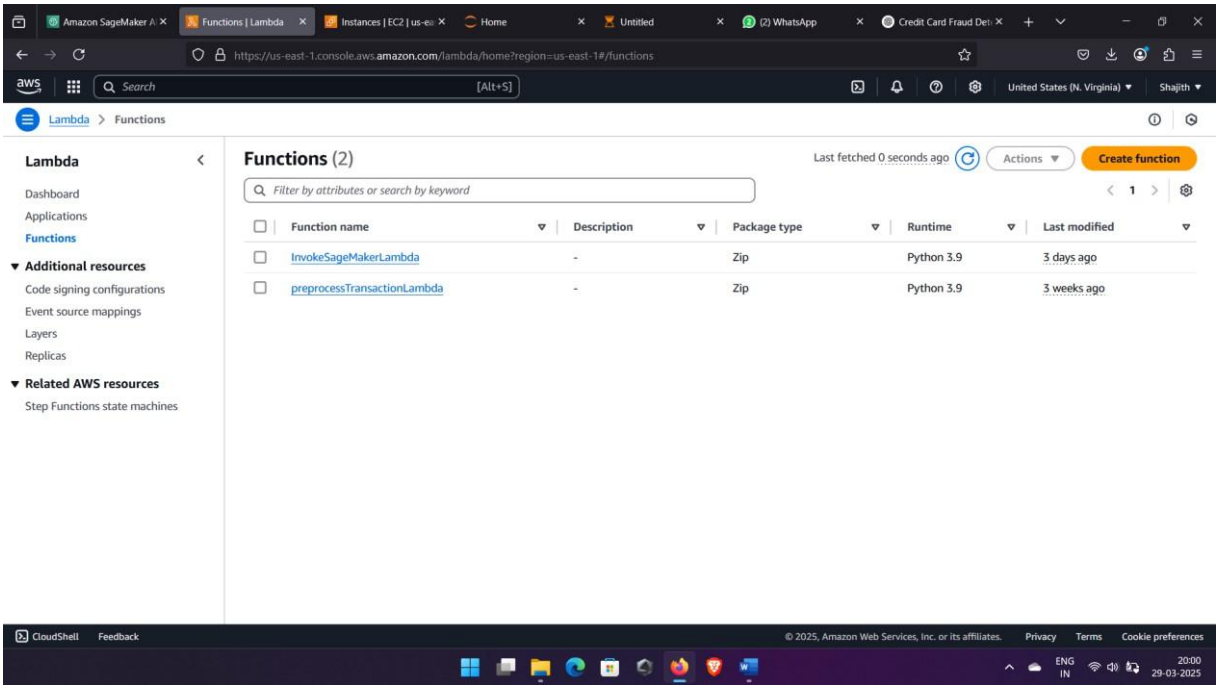


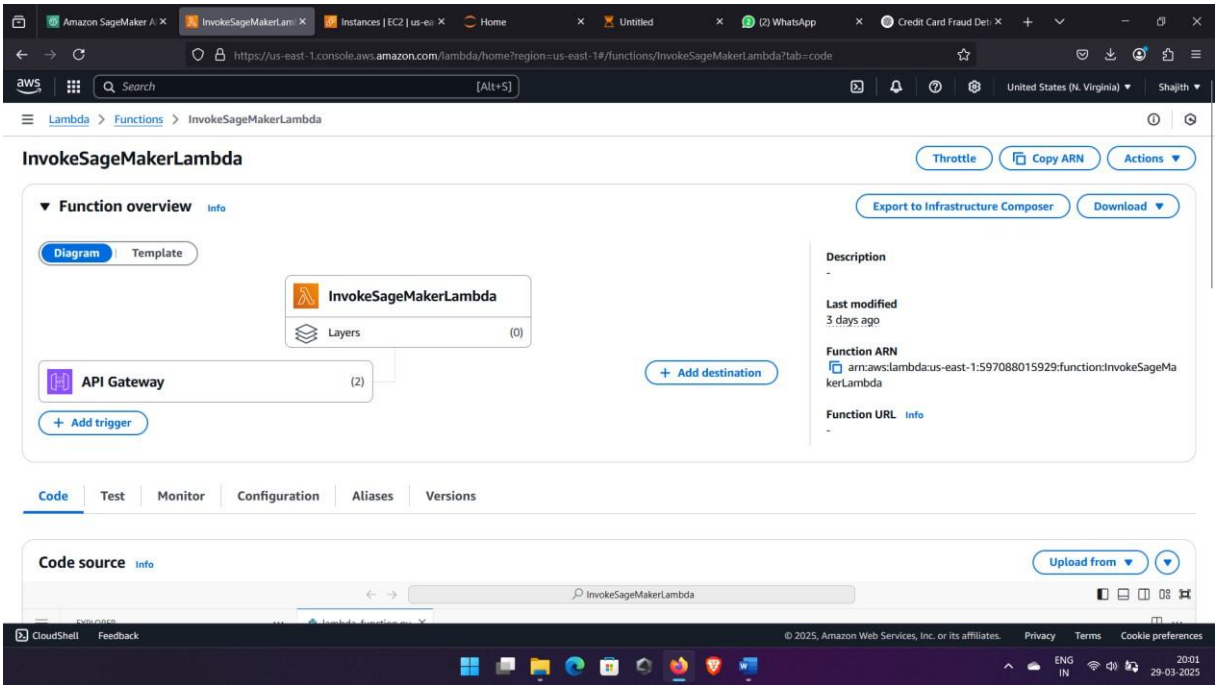
AWS Lambda

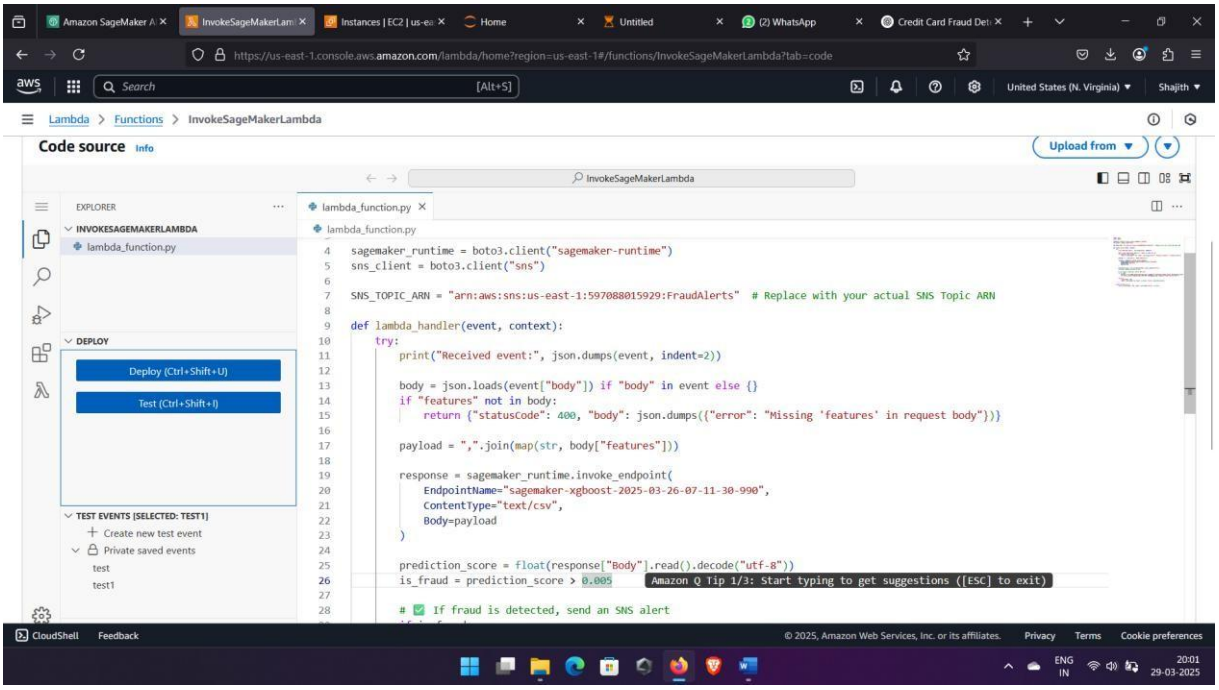
- Processes transaction data and invokes **SageMaker endpoint**.
- Sends alerts using **AWS SNS**.
- Logs details using **CloudWatch**.

Screenshot:

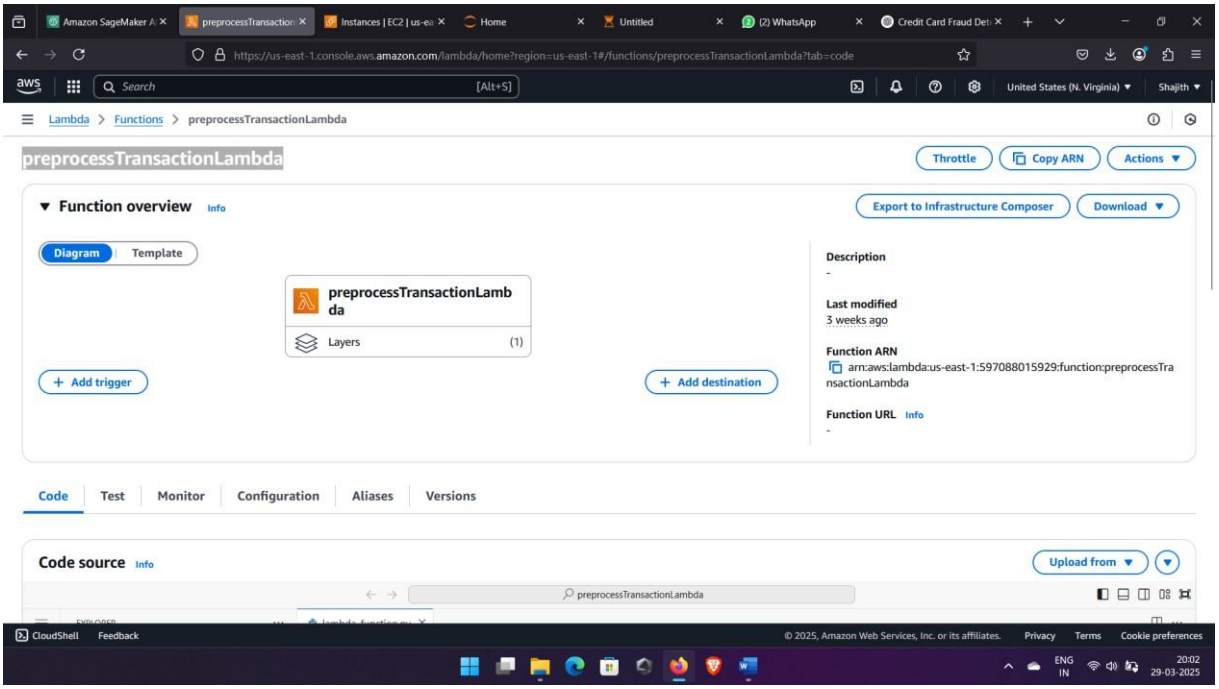
InvokeSageMakerLambda

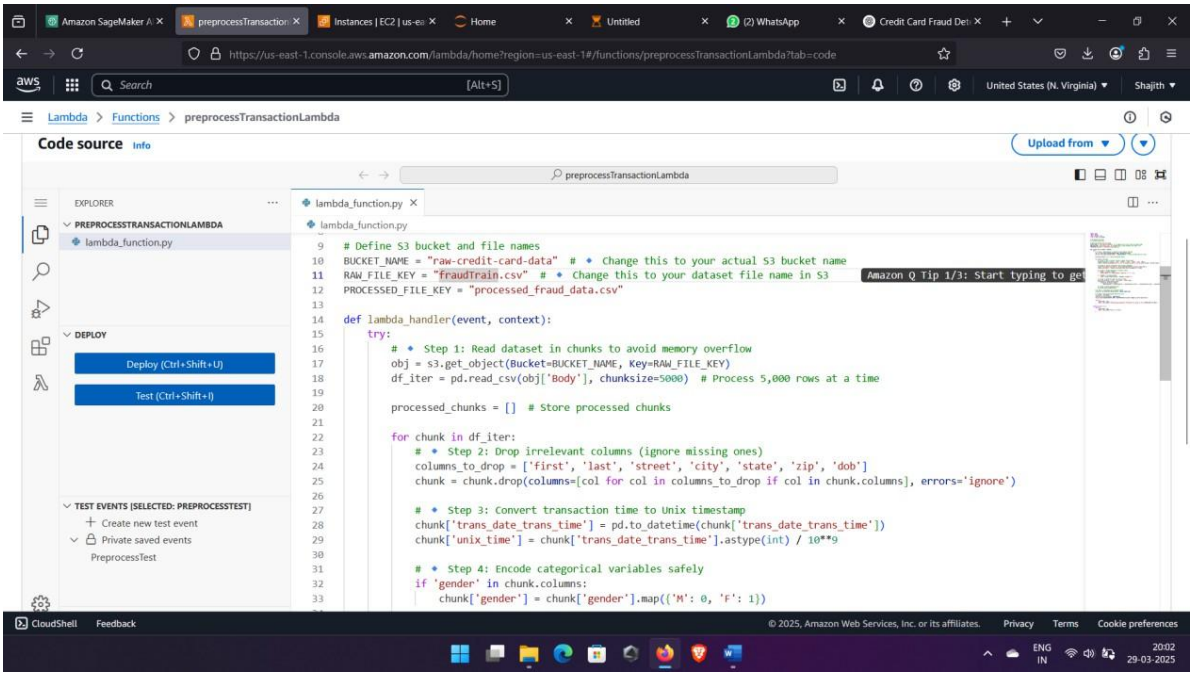


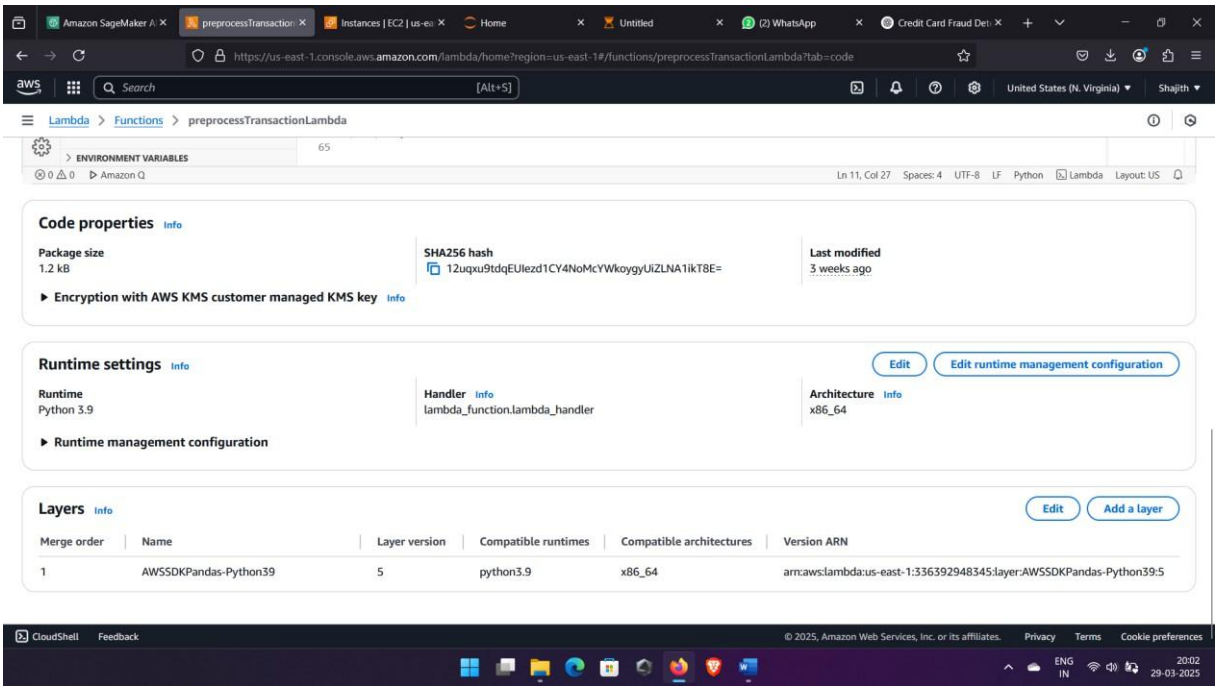

Lambda > Functions > InvokeSageMakerLambda
InvokeSageMakerLambda
Throttle
Copy ARN
Actions
Function overview
Info
Export to Infrastructure Composer
Download
Diagram
Template
InvokeSageMakerLambda
Layers
(0)
API Gateway
(2)
Add trigger
Add destination
Description
-
Last modified
3 days ago
Function ARN
arn:aws:lambda:us-east-1:597088015929:function:InvokeSageMakerLambda
Function URL
Info
-
Code
Test
Monitor
Configuration
Aliases
Versions
Code source
Info
Upload from
InvokeSageMakerLambda
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Lambda > Functions > InvokeSageMakerLambda
Code source
Info
Upload from
InvokeSageMakerLambda
EXPLORER
INVOKESAGEMAKERLAMBDA
lambda_function.py
DEPLOY
Deploy (Ctrl+Shift+U)
Test (Ctrl+Shift+I)
TEST EVENTS [SELECTED: TEST1]
Create new test event
Private saved events
test
test1
lambda_function.py
sagemaker_runtime = boto3.client("sagemaker-runtime")
sns_client = boto3.client("sns")
SNS_TOPIC_ARN = "arn:aws:sns:us-east-1:597088015929:FraudAlerts" # Replace with your actual SNS Topic ARN
def lambda_handler(event, context):
try:
print("Received event:", json.dumps(event, indent=2))
body = json.loads(event["body"]) if "body" in event else {}
if "features" not in body:
return {"statusCode": 400, "body": json.dumps({"error": "Missing 'features' in request body"})}
payload = ",".join(map(str, body["features"]))
response = sagemaker_runtime.invoke_endpoint(
EndpointName="sagemaker-xgboost-2025-03-26-07-11-30-990",
ContentType="text/csv",
Body=payload
)
prediction_score = float(response["Body"].read().decode("utf-8"))
is_fraud = prediction_score > 0.005
Amazon Q Tip 1/3: Start typing to get suggestions ([ESC] to exit)
If fraud is detected, send an SNS alert
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

preprocessTransactionLambda



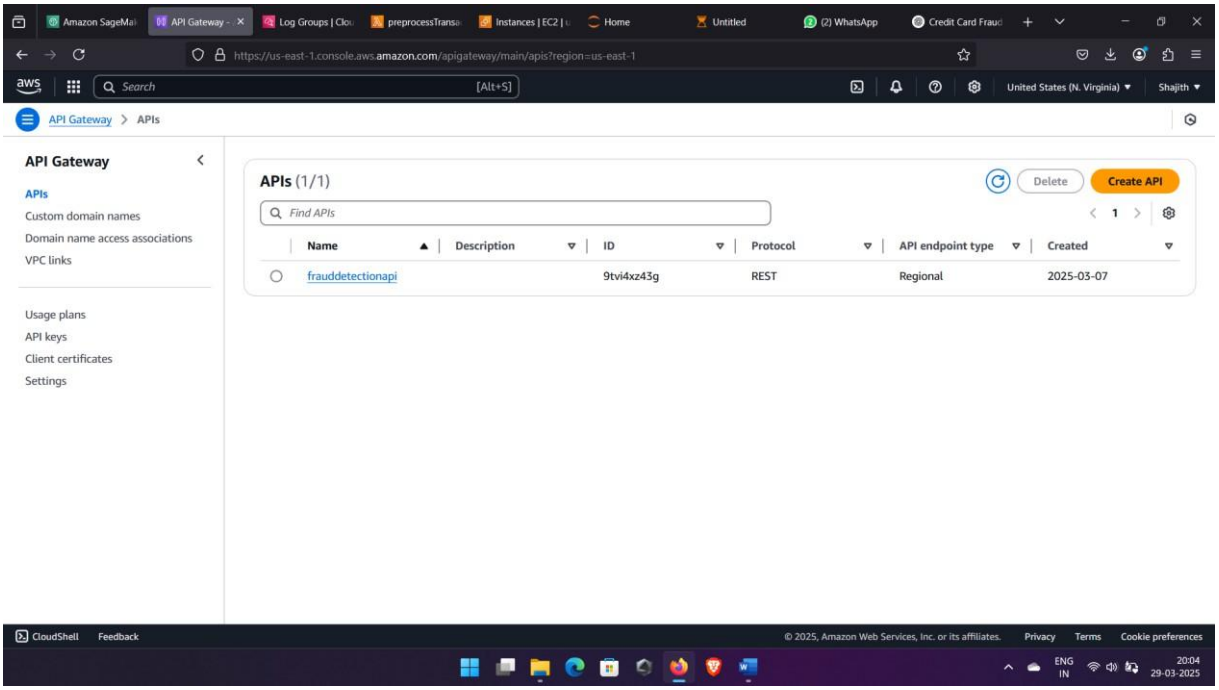


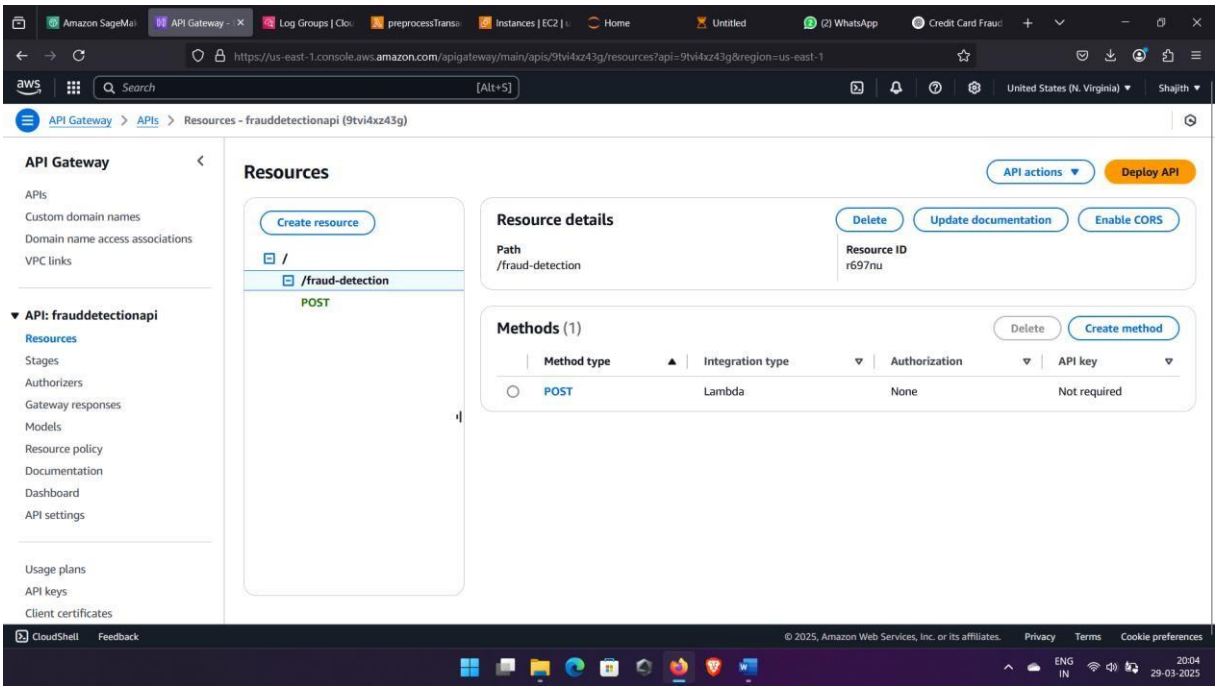


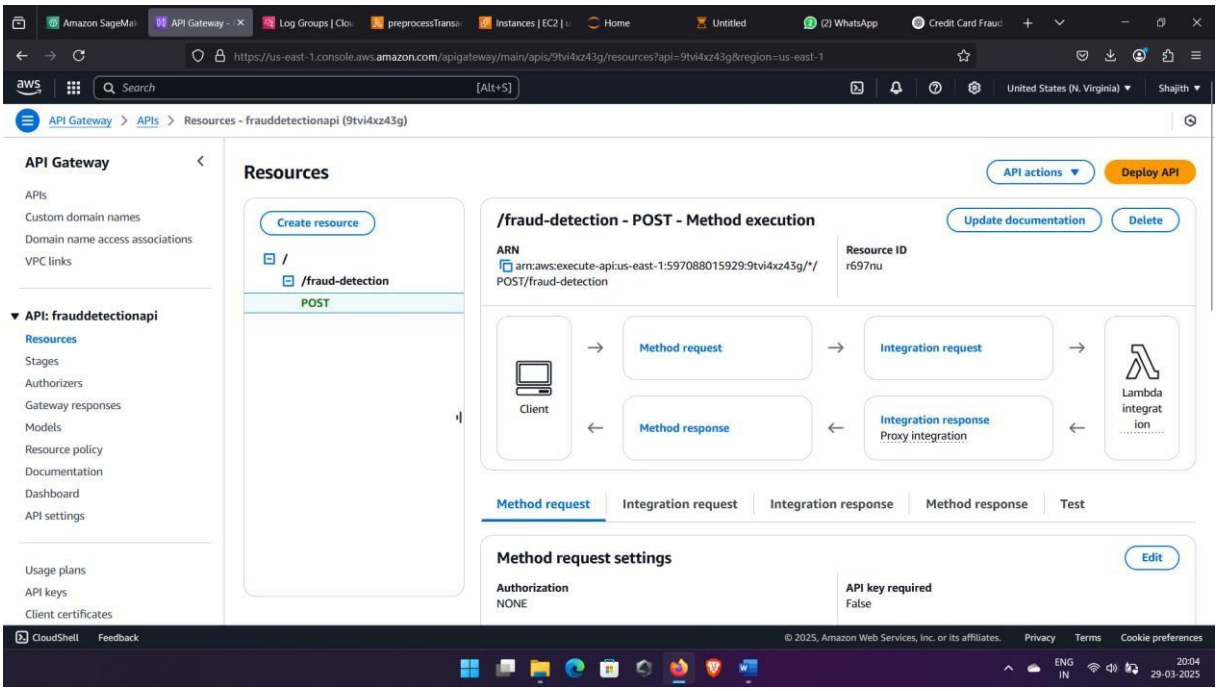
Amazon API Gateway

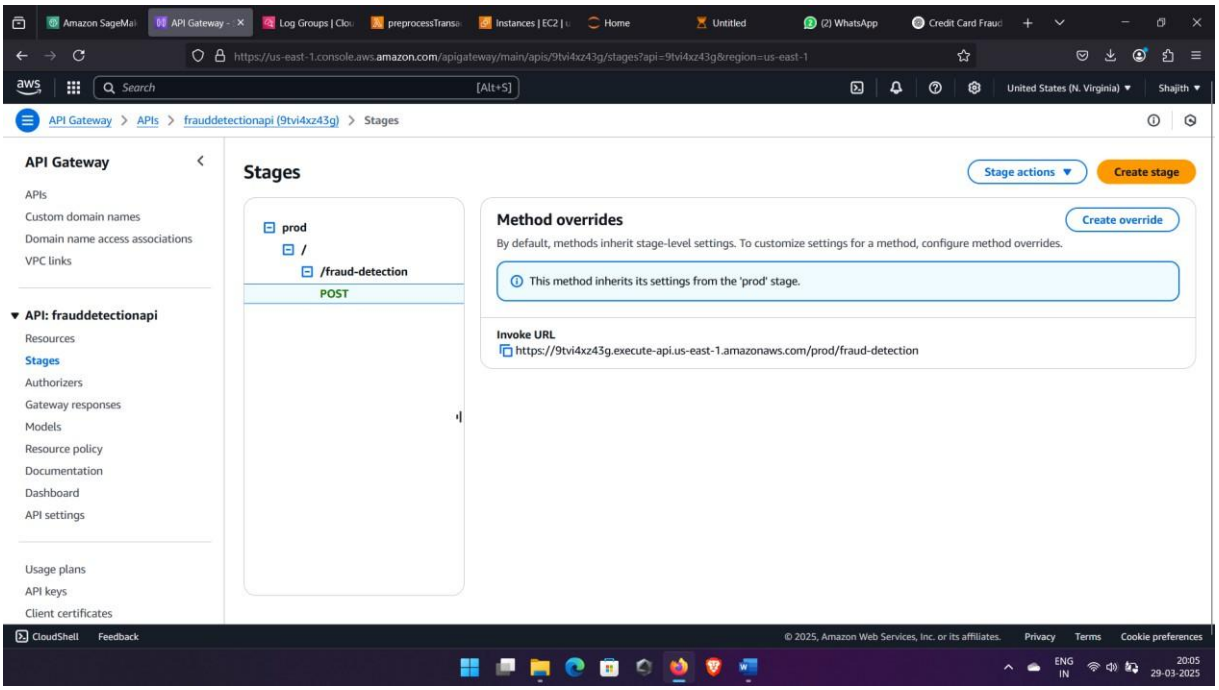
- Exposes a **REST API** endpoint for transaction requests.
- Secures the API with authentication mechanisms.

Screenshot:




API Gateway
APIs
Custom domain names
Domain name access associations
VPC links
API: frauddetectionapi
Resources
Stages
Authorizers
Gateway responses
Models
Resource policy
Documentation
Dashboard
API settings
Usage plans
API keys
Client certificates
Resources
API actions
Deploy API
Create resource
/
/fraud-detection
POST
Resource details
Delete
Update documentation
Enable CORS
Path
/fraud-detection
Resource ID
r697nu
Methods (1)
Delete
Create method
Method type
Integration type
Authorization
API key
POST
Lambda
None
Not required

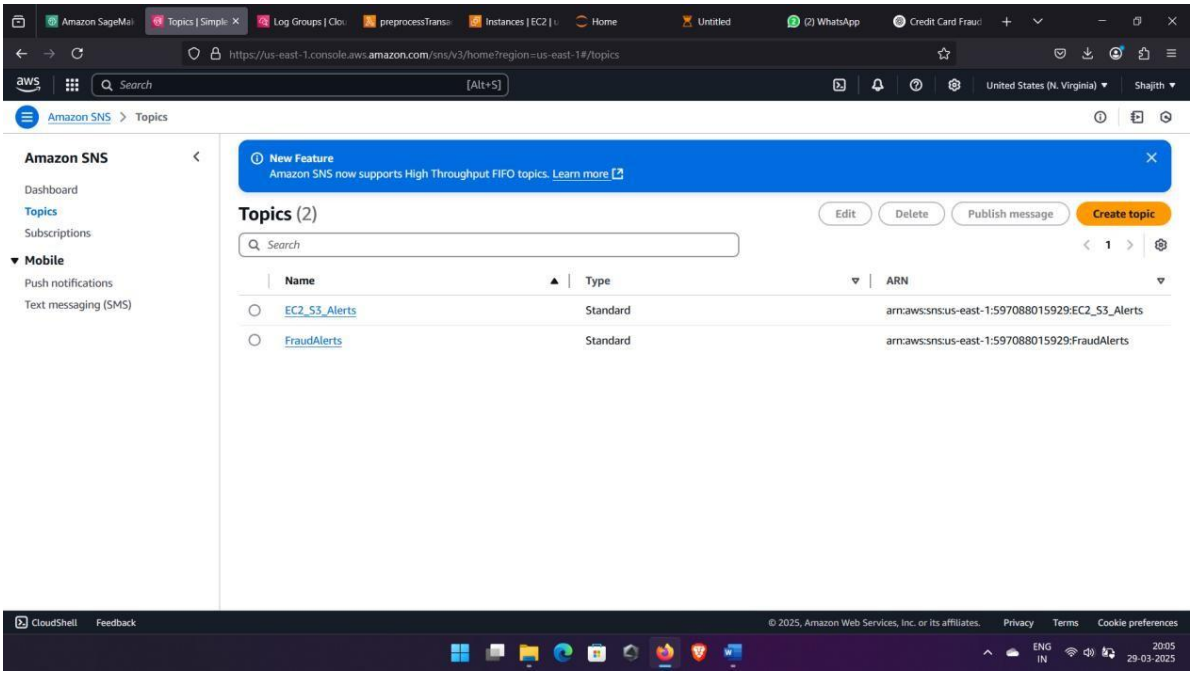

API Gateway
APIs
Custom domain names
Domain name access associations
VPC links
API: frauddetectionapi
Resources
Stages
Authorizers
Gateway responses
Models
Resource policy
Documentation
Dashboard
API settings
Usage plans
API keys
Client certificates
Resources
API actions
Deploy API
Create resource
/
/fraud-detection
POST
/fraud-detection - POST - Method execution
Update documentation
Delete
ARN
arn:aws:execute-api:us-east-1:597088015929:9tvi4xz43g/*/POST/fraud-detection
Resource ID
r697nu
Client
Method request
Integration request
Lambda integration
Method response
Integration response
Proxy integration
Method request
Integration request
Integration response
Method response
Test
Method request settings
Edit
Authorization
NONE
API key required
False

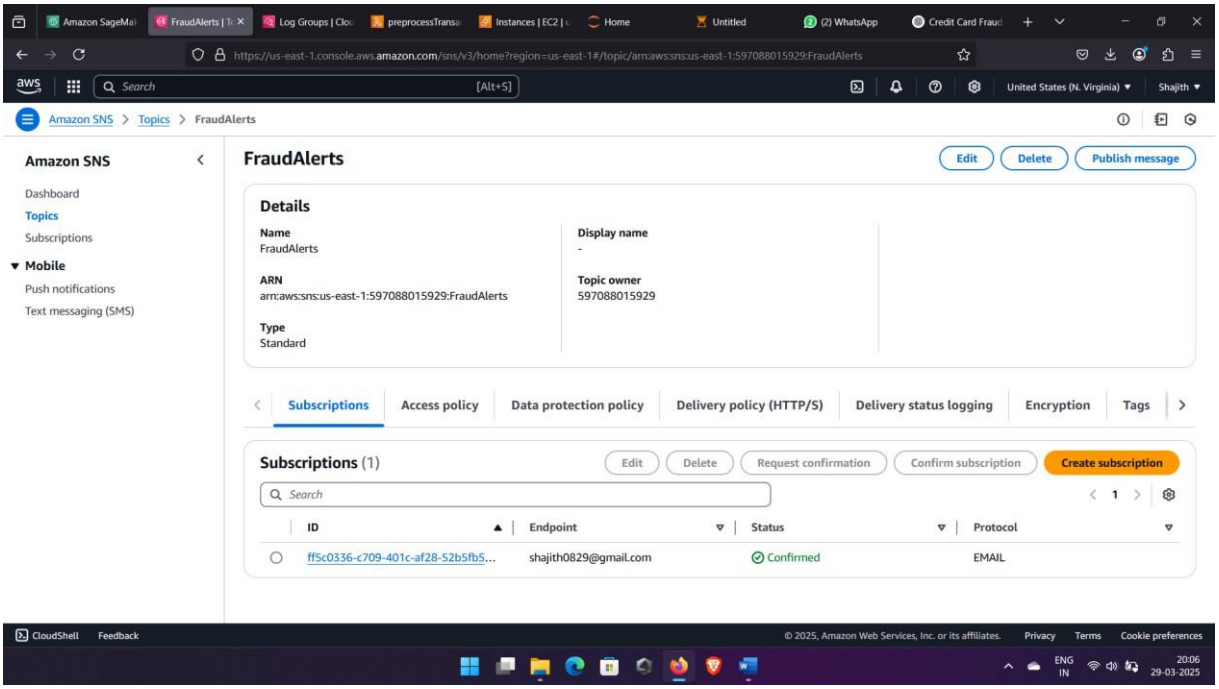


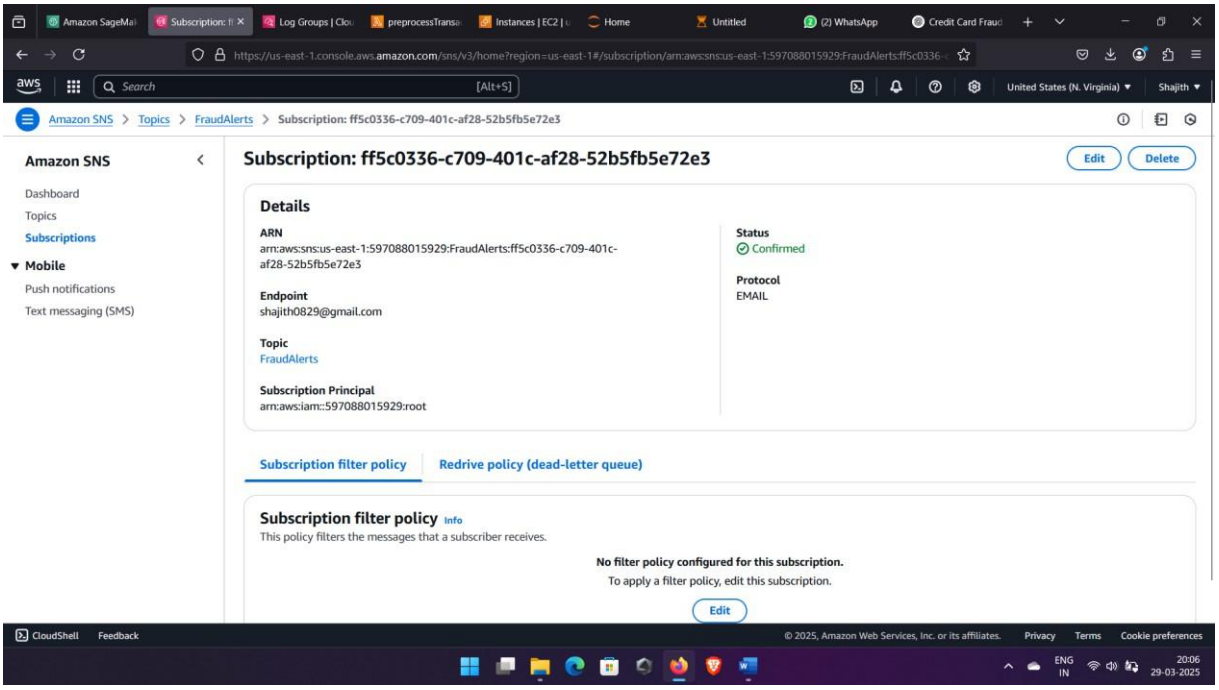
AWS SNS (Simple Notification Service)

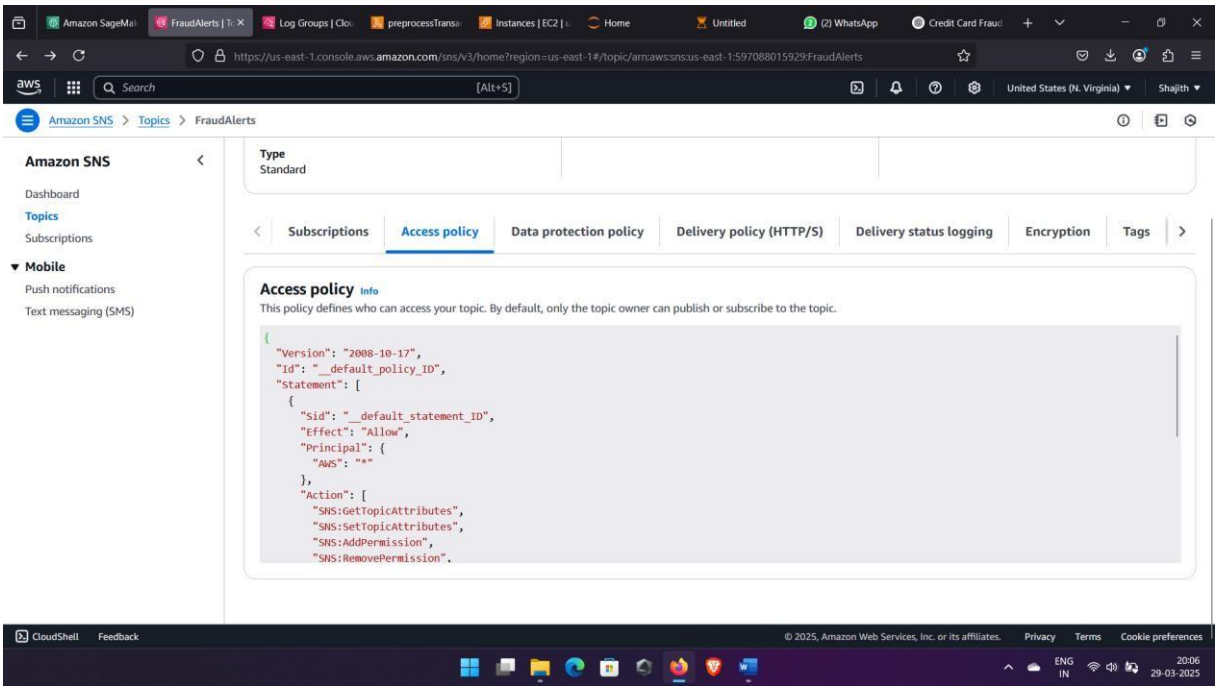
- Sends fraud alerts via email/SMS.

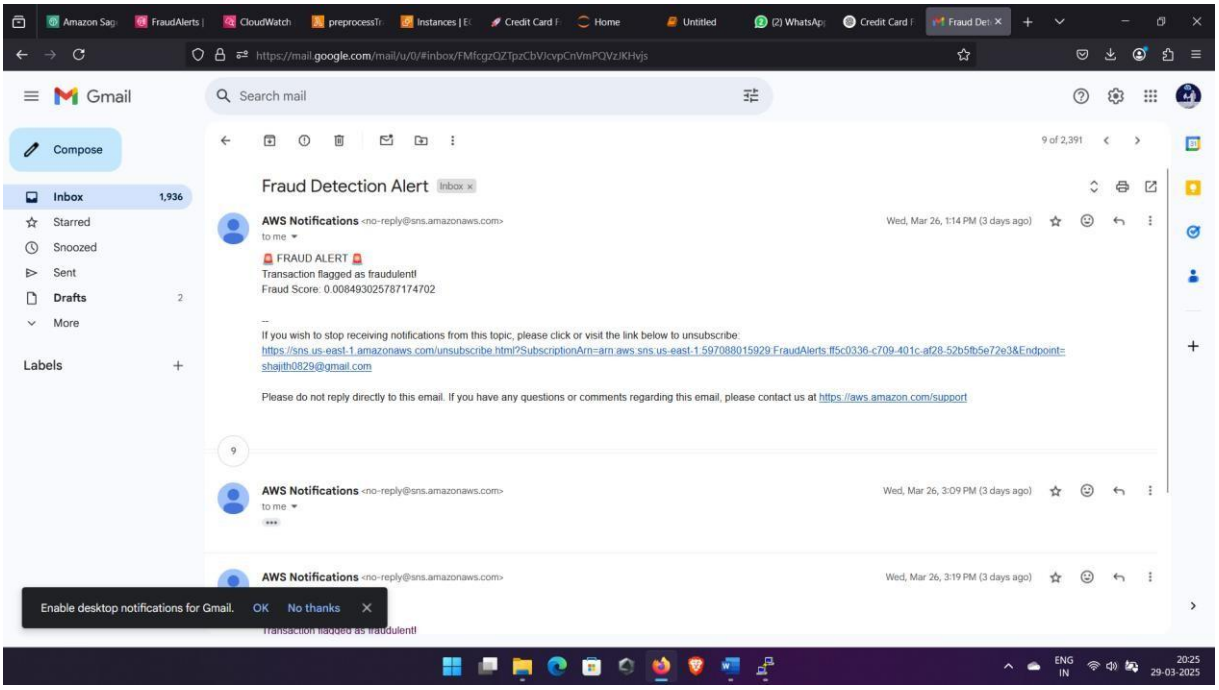
Screenshot:




Amazon SNS > Topics > FraudAlerts
Amazon SNS
Dashboard
Topics
Subscriptions
Mobile
Push notifications
Text messaging (SMS)
FraudAlerts
Edit
Delete
Publish message
Details
Name
FraudAlerts
Display name
-
ARN
arn:aws:sns:us-east-1:597088015929:FraudAlerts
Topic owner
597088015929
Type
Standard
Subscriptions
Access policy
Data protection policy
Delivery policy (HTTP/S)
Delivery status logging
Encryption
Tags
Subscriptions (1)
Edit
Delete
Request confirmation
Confirm subscription
Create subscription
Search
ID
Endpoint
Status
Protocol
ff5c0336-c709-401c-af28-52b5fb5...
shajith0829@gmail.com
Confirmed
EMAIL
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Amazon SNS > Topics > FraudAlerts > Subscription: ff5c0336-c709-401c-af28-52b5fb5e72e3
Amazon SNS
Dashboard
Topics
Subscriptions
Mobile
Push notifications
Text messaging (SMS)
Subscription: ff5c0336-c709-401c-af28-52b5fb5e72e3
Edit
Delete
Details
ARN
arn:aws:sns:us-east-1:597088015929:FraudAlerts:ff5c0336-c709-401c-af28-52b5fb5e72e3
Status
Confirmed
Protocol
EMAIL
Endpoint
shajith0829@gmail.com
Topic
FraudAlerts
Subscription Principal
arn:aws:iam::597088015929:root
Subscription filter policy
Redrive policy (dead-letter queue)
Subscription filter policy Info
This policy filters the messages that a subscriber receives.
No filter policy configured for this subscription.
To apply a filter policy, edit this subscription.
Edit
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

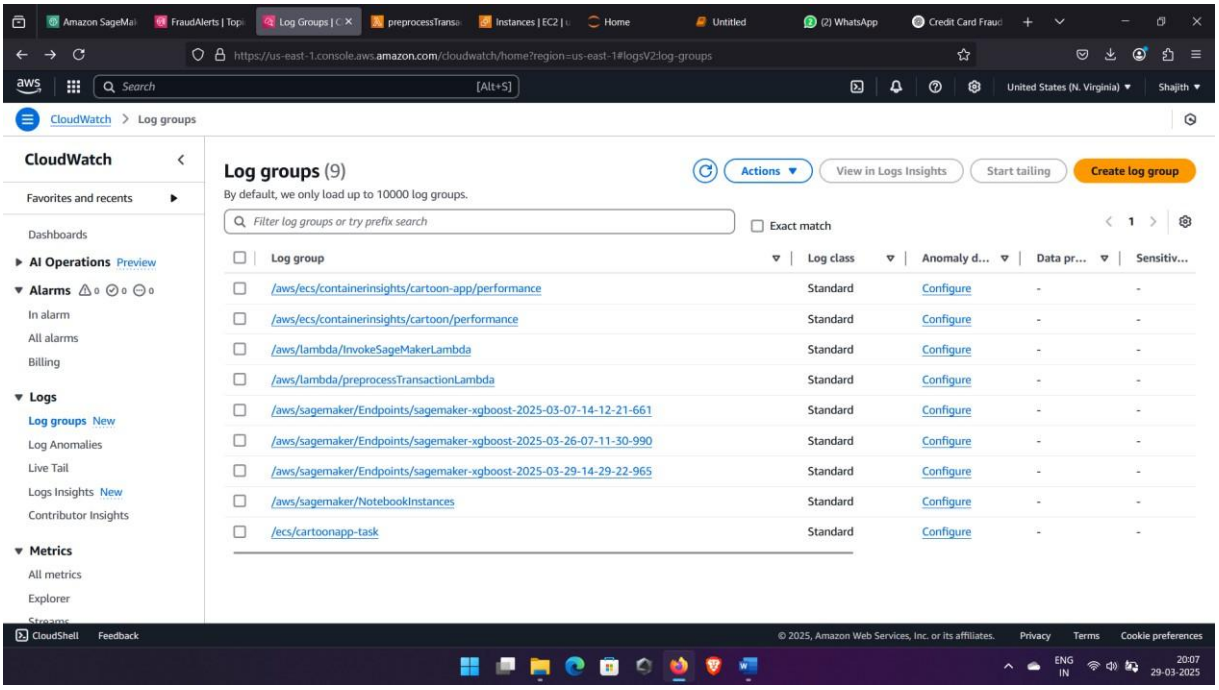




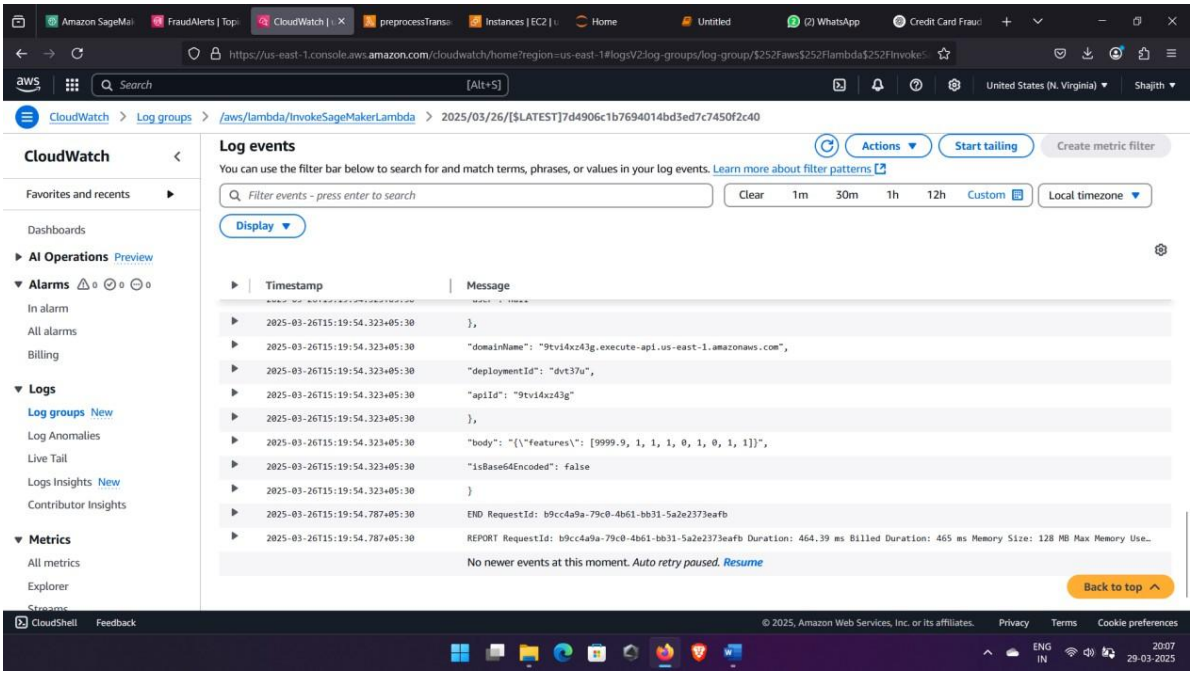
AWS CloudWatch

- Monitors API calls, Lambda execution logs, and SNS alerts.
- Detects unusual spikes in fraud attempts.

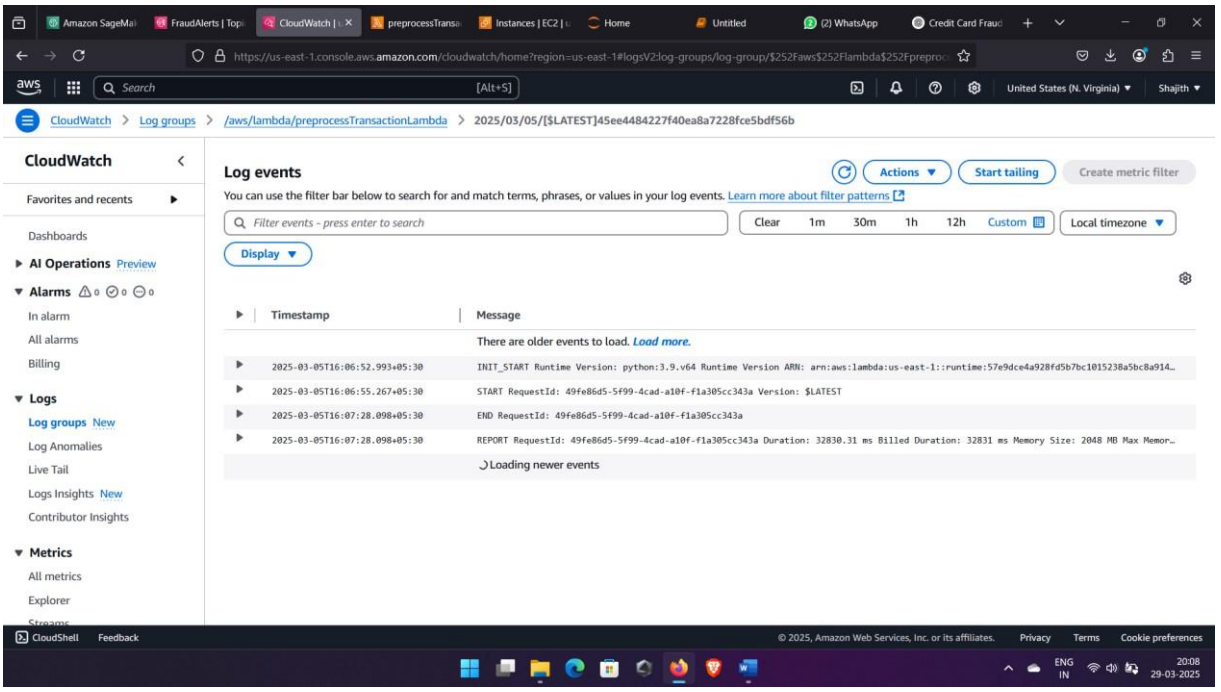
Screenshot:



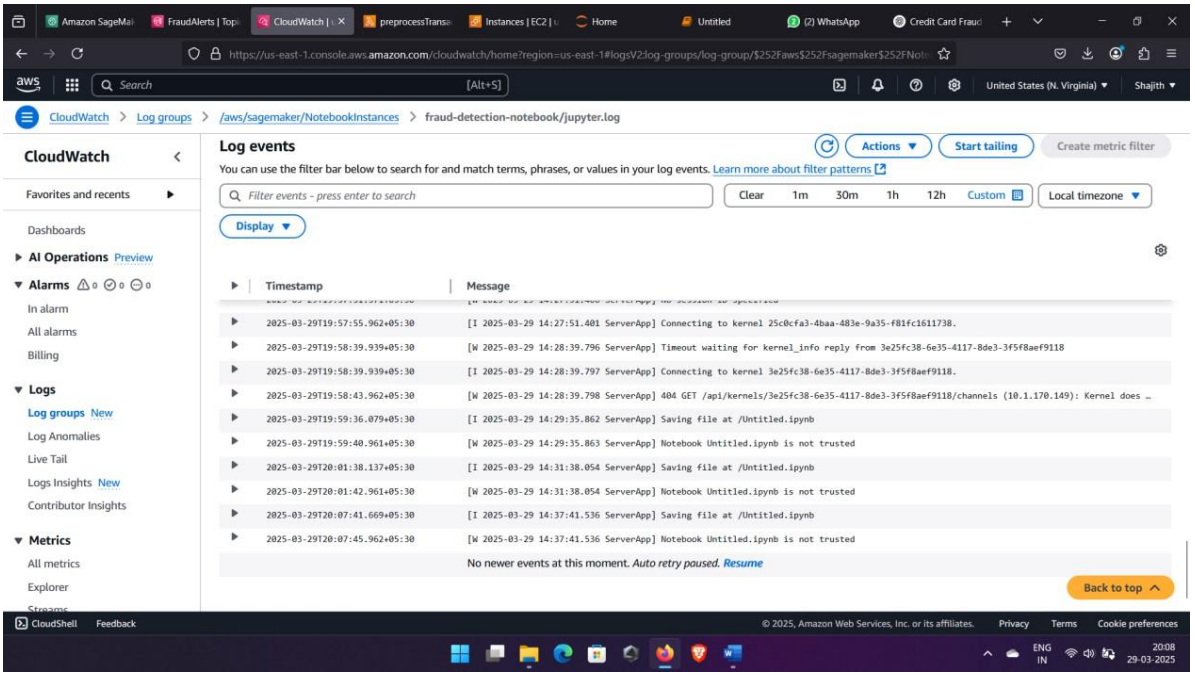
Lambda Log:



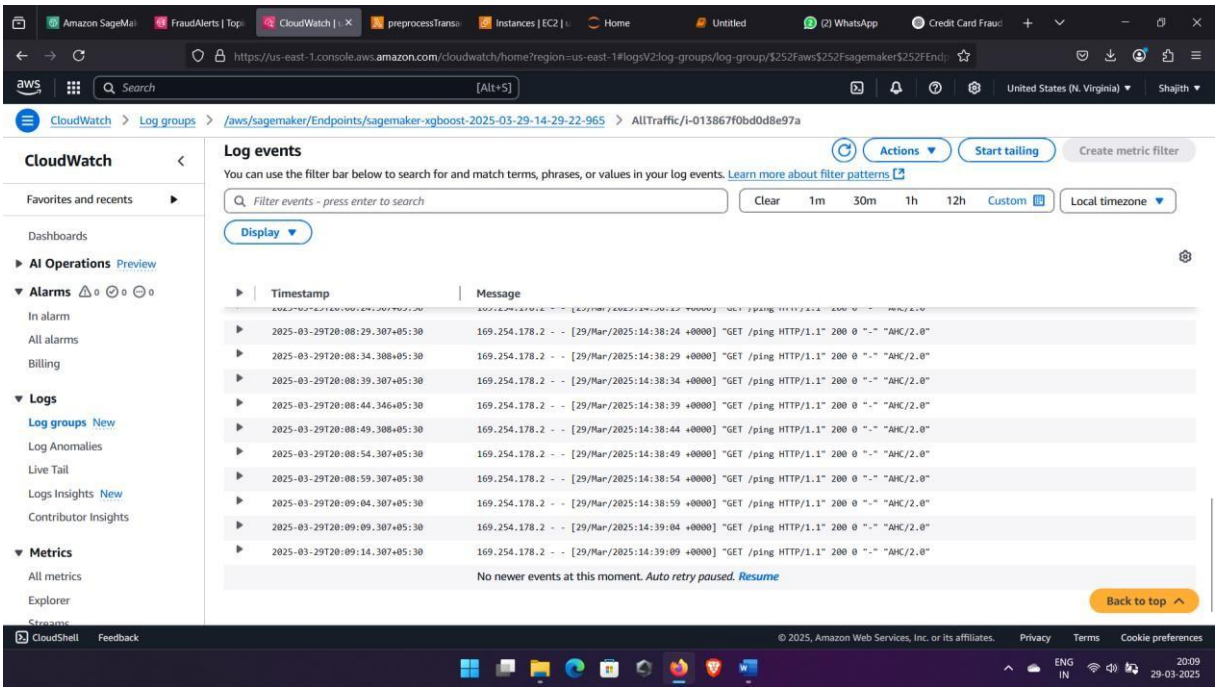
Preprocessing Log:



Sagemaker Jupter Notebook Log:



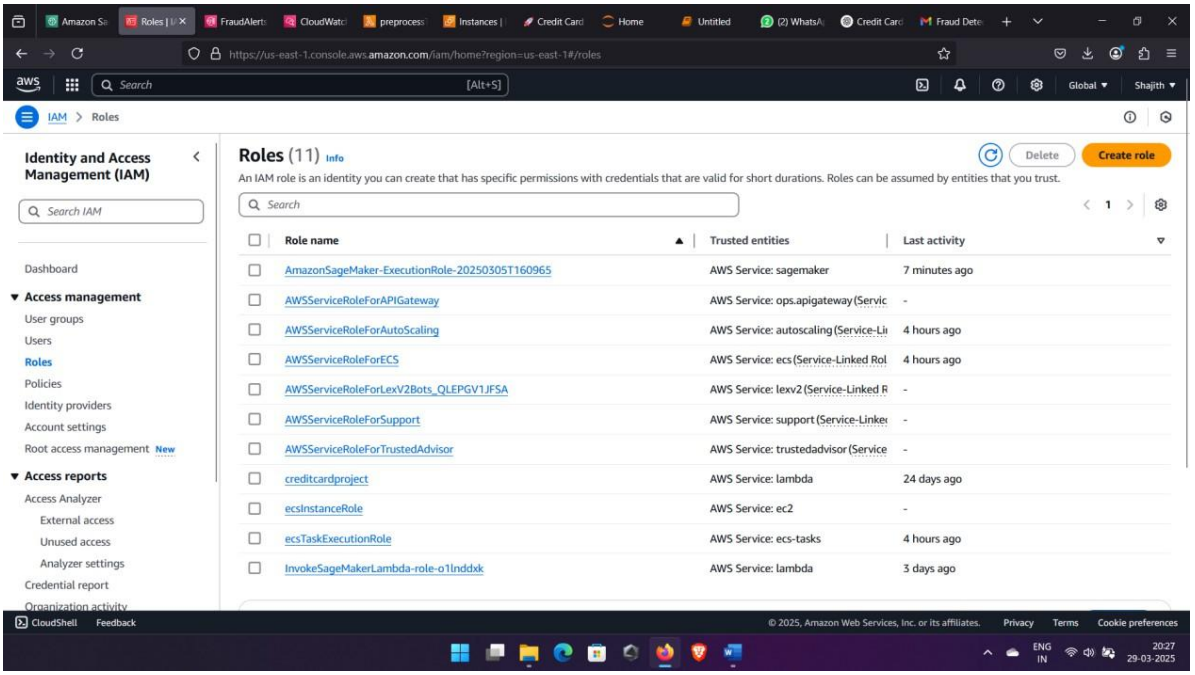
Sagemaker Endpoint Logs:

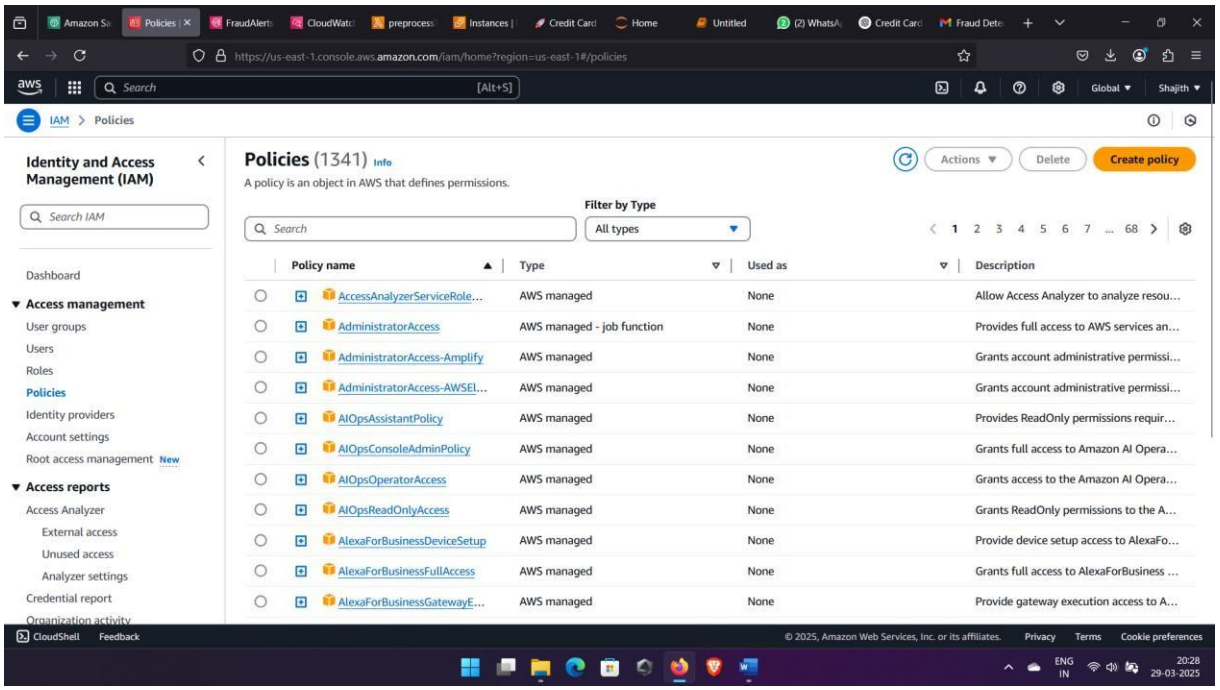


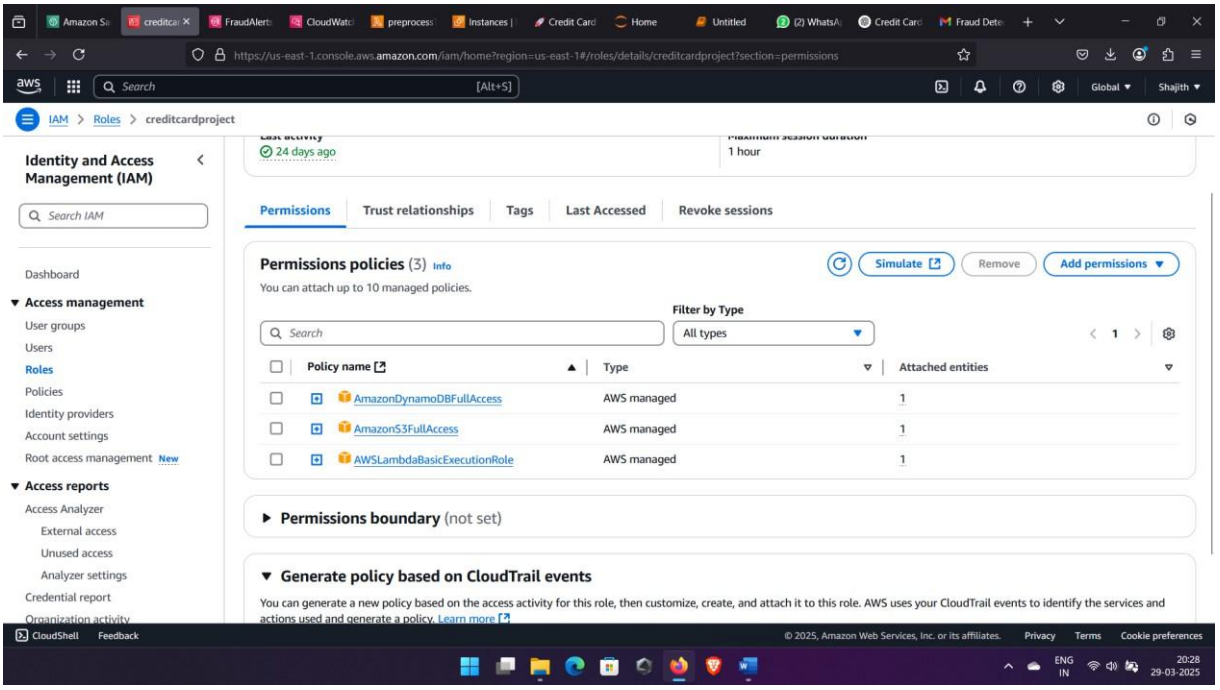
AWS IAM (Identity and Access Management)

- Restricts access to SageMaker, Lambda, and API Gateway.
- Ensures secure API calls.

Screenshot:



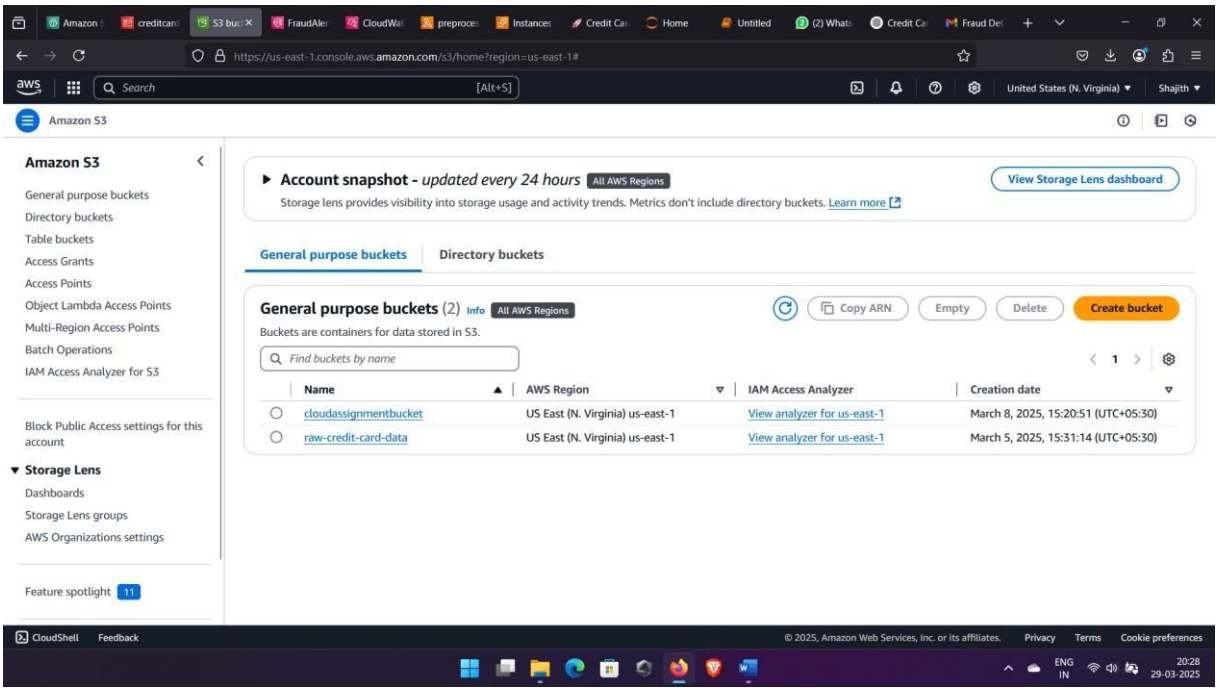


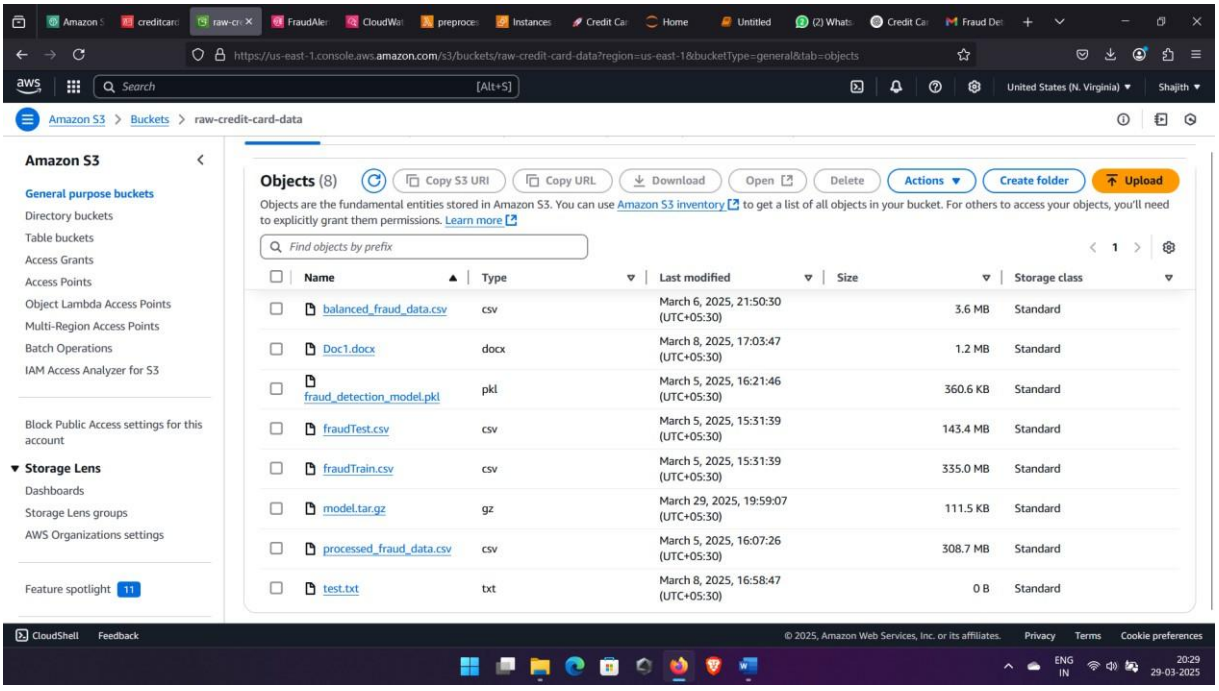


Amazon S3

- Stores datasets for fraud detection model training.
- Stores preprocessed transaction data from Lambda.

Screenshot:

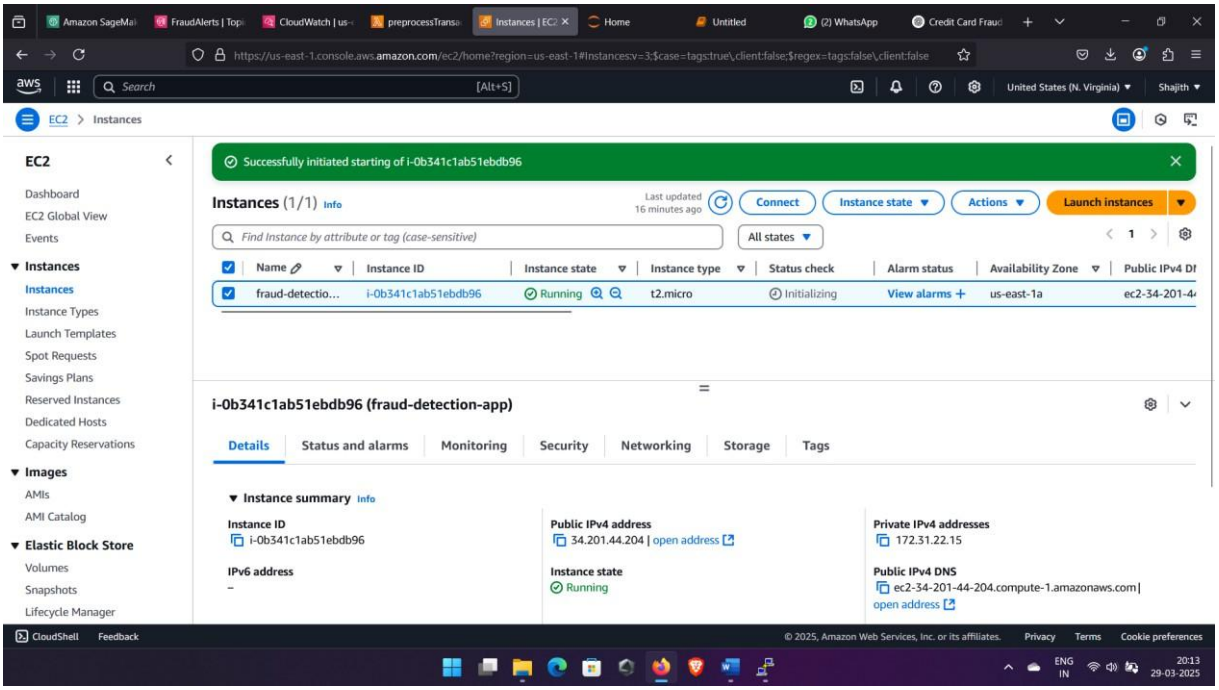


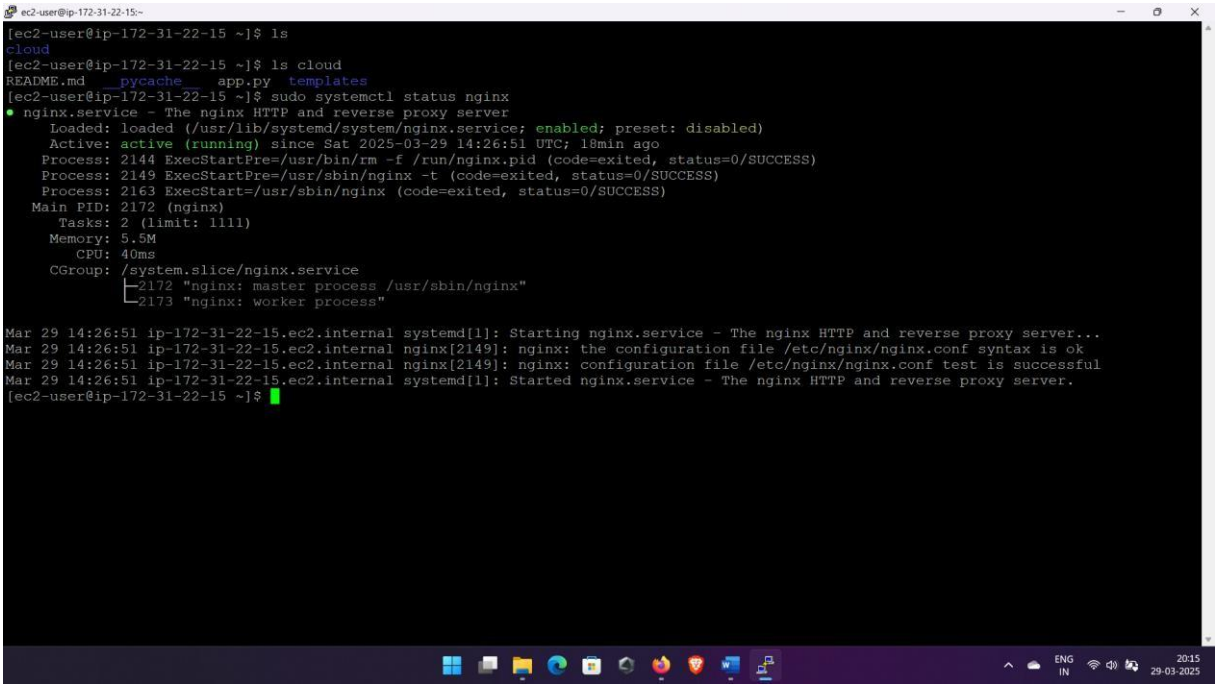


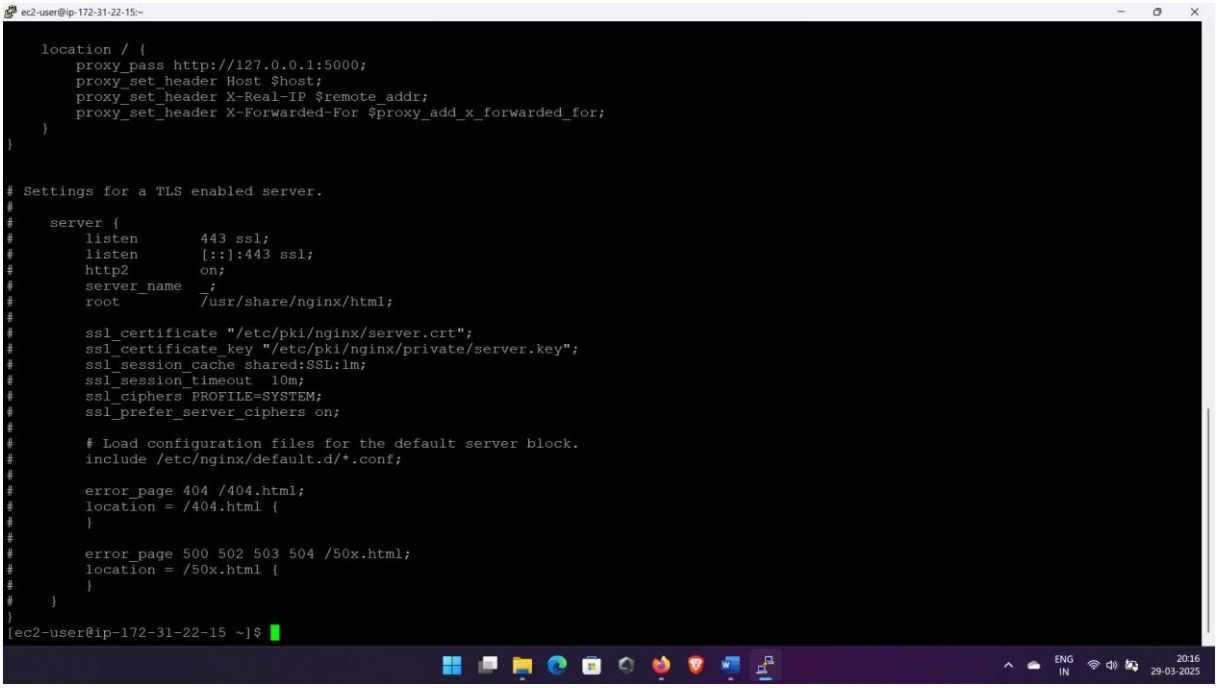
AWS EC2

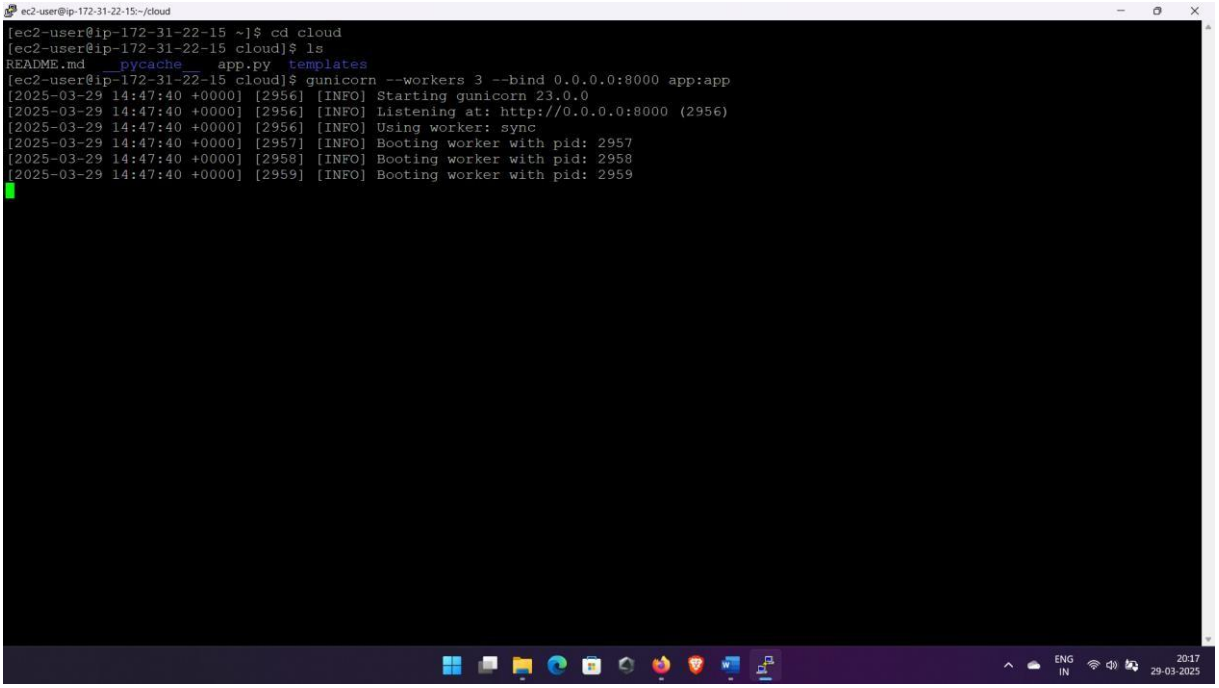
- Hosts Flask-based UI for user transactions.
- Runs Nginx and Gunicorn for backend support.

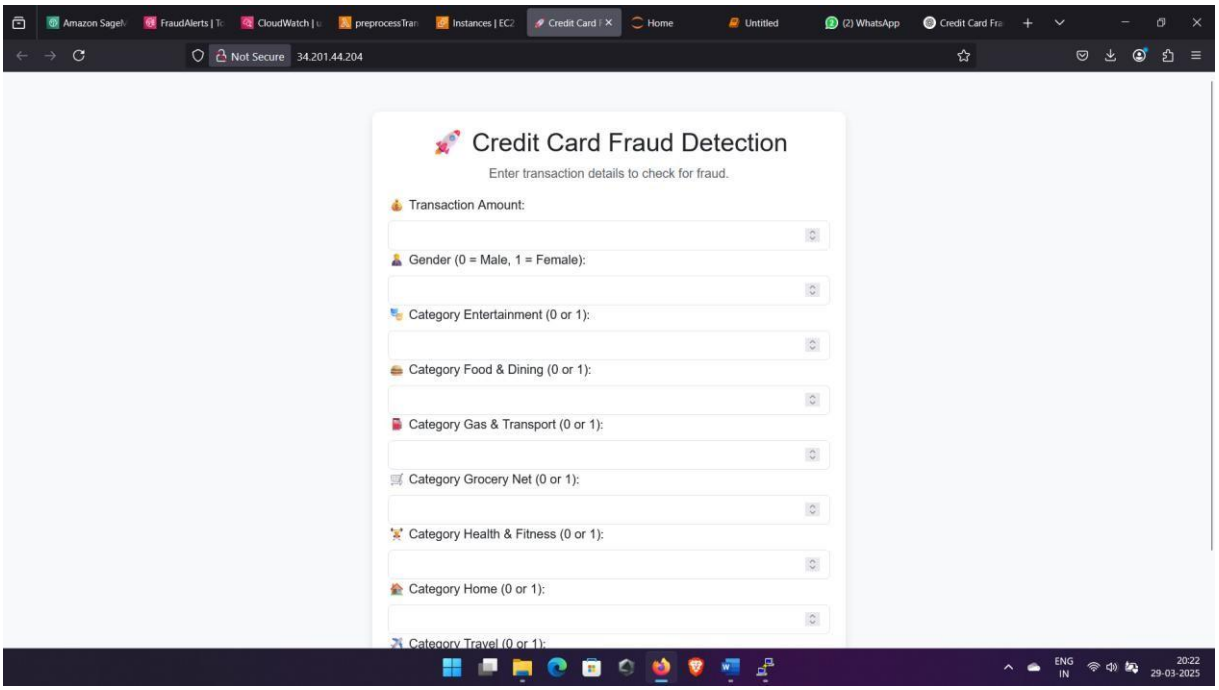
Screenshot:

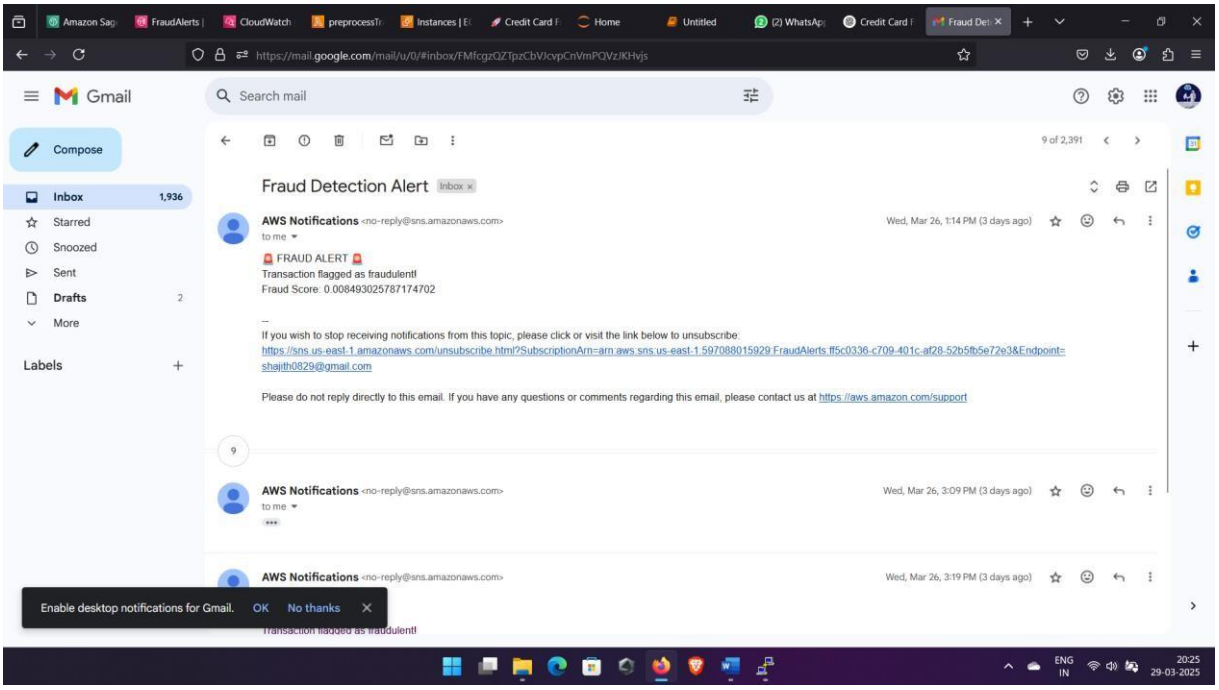

Successfully initiated starting of i-0b341c1ab51ebdb96
Instances (1/1) Info
Connect
Instance state
Actions
Launch instances
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
fraud-detectio...
i-0b341c1ab51ebdb96
Running
t2.micro
Initializing
View alarms +
us-east-1a
i-0b341c1ab51ebdb96 (fraud-detection-app)
Details
Status and alarms
Monitoring
Security
Networking
Storage
Tags
Instance summary Info
Instance ID
i-0b341c1ab51ebdb96
Public IPv4 address
34.201.44.204 | open address
Private IPv4 addresses
172.31.22.15
IPv6 address
Instance state
Running
Public IPv4 DNS
ec2-34-201-44-204.compute-1.amazonaws.com |
open address
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


[ec2-user@ip-172-31-22-15 ~]$ ls
cloud
[ec2-user@ip-172-31-22-15 ~]$ ls cloud
README.md __pycache__ app.py templates
[ec2-user@ip-172-31-22-15 ~]$ sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disabled)
Active: active (running) since Sat 2025-03-29 14:26:51 UTC; 18min ago
Process: 2144 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
Process: 2149 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
Process: 2163 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
Main PID: 2172 (nginx)
Tasks: 2 (limit: 1111)
Memory: 5.5M
CPU: 40ms
CGroup: /system.slice/nginx.service
├─2172 "nginx: master process /usr/sbin/nginx"
└─2173 "nginx: worker process"
Mar 29 14:26:51 ip-172-31-22-15.ec2.internal systemd[1]: Starting nginx.service - The nginx HTTP and reverse proxy server...
Mar 29 14:26:51 ip-172-31-22-15.ec2.internal nginx[2149]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Mar 29 14:26:51 ip-172-31-22-15.ec2.internal nginx[2149]: nginx: configuration file /etc/nginx/nginx.conf test is successful
Mar 29 14:26:51 ip-172-31-22-15.ec2.internal systemd[1]: Started nginx.service - The nginx HTTP and reverse proxy server.
[ec2-user@ip-172-31-22-15 ~]$


ec2-user@ip-172-31-22-15:~
location / {
proxy_pass http://127.0.0.1:5000;
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}
}
# Settings for a TLS enabled server.
#
server {
listen 443 ssl;
listen [::]:443 ssl;
http2 on;
server_name _;
root /usr/share/nginx/html;
#
ssl_certificate "/etc/pki/nginx/server.crt";
ssl_certificate_key "/etc/pki/nginx/private/server.key";
ssl_session_cache shared:SSL:1m;
ssl_session_timeout 10m;
ssl_ciphers PROFILE=SYSTEM;
ssl_prefer_server_ciphers on;
#
Load configuration files for the default server block.
include /etc/nginx/default.d/*.conf;
#
error_page 404 /404.html;
location = /404.html {
}
#
error_page 500 502 503 504 /50x.html;
location = /50x.html {
}
}
}
[ec2-user@ip-172-31-22-15 ~]$
ENG
IN
20:16
29-03-2025


ec2-user@ip-172-31-22-15:~/cloud
[ec2-user@ip-172-31-22-15 ~]$ cd cloud
[ec2-user@ip-172-31-22-15 cloud]$ ls
README.md __pycache__ app.py templates
[ec2-user@ip-172-31-22-15 cloud]$ gunicorn --workers 3 --bind 0.0.0.0:8000 app:app
[2025-03-29 14:47:40 +0000] [2956] [INFO] Starting gunicorn 23.0.0
[2025-03-29 14:47:40 +0000] [2956] [INFO] Listening at: http://0.0.0.0:8000 (2956)
[2025-03-29 14:47:40 +0000] [2956] [INFO] Using worker: sync
[2025-03-29 14:47:40 +0000] [2957] [INFO] Booting worker with pid: 2957
[2025-03-29 14:47:40 +0000] [2958] [INFO] Booting worker with pid: 2958
[2025-03-29 14:47:40 +0000] [2959] [INFO] Booting worker with pid: 2959
ENG
IN
20:17
29-03-2025


Not Secure 34.201.44.204
Credit Card Fraud Detection
Enter transaction details to check for fraud.
Transaction Amount:
Gender (0 = Male, 1 = Female):
Category Entertainment (0 or 1):
Category Food & Dining (0 or 1):
Category Gas & Transport (0 or 1):
Category Grocery Net (0 or 1):
Category Health & Fitness (0 or 1):
Category Home (0 or 1):
Category Travel (0 or 1):


https://mail.google.com/mail/u/0/#inbox/FMfcgzQZTpzCbVJcvpCnVmPQVzJKHvjs
Gmail
Search mail
Compose
Inbox 1,936
Starred
Snoozed
Sent
Drafts 2
More
Labels
Fraud Detection Alert
AWS Notifications <no-reply@sns.amazonaws.com>
Wed, Mar 26, 1:14 PM (3 days ago)
FRAUD ALERT
Transaction flagged as fraudulent!
Fraud Score: 0.008493025787174702
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:597088015929:FraudAlerts:ff5c0336-c709-401c-af28-52b5fb5e72e3&Endpoint=shajith0829@gmail.com
Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support
AWS Notifications <no-reply@sns.amazonaws.com>
Wed, Mar 26, 3:09 PM (3 days ago)
AWS Notifications <no-reply@sns.amazonaws.com>
Wed, Mar 26, 3:19 PM (3 days ago)
Enable desktop notifications for Gmail. OK No thanks

4. Coding:

Lambda:

InvokeSageMakerLambda

```
import json
import boto3

sagemaker_runtime = boto3.client("sagemaker-runtime")
sns_client = boto3.client("sns")

SNS_TOPIC_ARN = "arn:aws:sns:us-east-1:597088015929:FraudAlerts"

def lambda_handler(event, context):
    try:
        print("Received event:", json.dumps(event, indent=2))

        body = json.loads(event["body"]) if "body" in event else {}
        if "features" not in body:
            return {"statusCode": 400, "body": json.dumps({"error": "Missing 'features' in request body"})}

        payload = ",".join(map(str, body["features"]))

        response = sagemaker_runtime.invoke_endpoint(
            EndpointName="sagemaker-xgboost-2025-03-26-07-11-30-990",
            ContentType="text/csv",
            Body=payload
        )

        prediction_score = float(response["Body"].read().decode("utf-8"))
        is_fraud = prediction_score > 0.005

        if is_fraud:
            message = f"🚨 FRAUD ALERT 🚨 \nTransaction flagged as fraudulent!\nFraud Score: {prediction_score}"
            sns_client.publish(TopicArn=SNS_TOPIC_ARN, Message=message, Subject="Fraud Detection Alert")

        return {
            "statusCode": 200,
            "body": json.dumps({"is_fraud": is_fraud, "score": prediction_score})
        }

    except Exception as e:
        return {"statusCode": 500, "body": json.dumps({"error": str(e)})}
```

preprocessTransactionLambda:

```
import json
import boto3
import pandas as pd
from io import StringIO

s3 = boto3.client('s3')

BUCKET_NAME = "raw-credit-card-data"
RAW_FILE_KEY = "fraudTrain.csv"
PROCESSED_FILE_KEY = "processed_fraud_data.csv"

def lambda_handler(event, context):
    try:

        obj = s3.get_object(Bucket=BUCKET_NAME, Key=RAW_FILE_KEY)
        df_iter = pd.read_csv(obj['Body'], chunksize=5000)

        processed_chunks = []

        for chunk in df_iter:

            columns_to_drop = ['first', 'last', 'street', 'city', 'state', 'zip', 'dob']
            chunk = chunk.drop(columns=[col for col in columns_to_drop if col in
chunk.columns], errors='ignore')


            chunk['trans_date_trans_time'] = pd.to_datetime(chunk['trans_date_trans_time'])
            chunk['unix_time'] = chunk['trans_date_trans_time'].astype(int) / 10**9


            if 'gender' in chunk.columns:
                chunk['gender'] = chunk['gender'].map({'M': 0, 'F': 1})

            if 'category' in chunk.columns:
                chunk = pd.get_dummies(chunk, columns=['category'])


            numerical_features = ['amt', 'merch_lat', 'merch_long']
            for feature in numerical_features:
                if feature in chunk.columns:
                    chunk[feature] = (chunk[feature] - chunk[feature].min()) /
(chunk[feature].max() - chunk[feature].min())

            processed_chunks.append(chunk)

        final_df = pd.concat(processed_chunks, ignore_index=True)
```

```
        csv_buffer = StringIO()
        final_df.to_csv(csv_buffer, index=False)
        s3.put_object(Bucket=BUCKET_NAME, Key=PROCESSED_FILE_KEY,
Body=csv_buffer.getvalue())

        return {
            'statusCode': 200,
            'body': json.dumps(f'Preprocessing Complete! Processed file saved in S3 as
{PROCESSED_FILE_KEY}')
        }

    except Exception as e:
        return {
            'statusCode': 500,
            'body': json.dumps(f"Error: {str(e)}")
        }
```

5. Conclusion

This project successfully implements real-time credit card fraud detection using **AWS cloud services**. The system is capable of accurately identifying fraudulent transactions, alerting users through SNS, and ensuring secure transaction monitoring via CloudWatch and IAM policies. The project demonstrates how AWS services can be effectively integrated to build a scalable, high-performance fraud detection system.

Furthermore, this system provides real-time fraud prevention capabilities that enhance the security of financial transactions. Future improvements will focus on:

- Enhancing machine learning model **accuracy** by training with larger datasets.
- Implementing real-time dashboards using AWS QuickSight for better analytics.
- Utilizing Auto Scaling to dynamically adjust system resources.
- Strengthening **security measures** through enhanced IAM policies and encryption.

By leveraging AWS's cloud-based architecture, this project provides a cost-efficient**,** scalable, and effective fraud detection system that can be further expanded to meet evolving security challenges in the financial industry.

References:

[1] Amazon Web Services (AWS) Documentation - https://docs.aws.amazon.com/

[2] Machine Learning for Fraud Detection, AWS Blog - https://aws.amazon.com/blogs/

[3] Real-time Analytics with Amazon Kinesis - https://aws.amazon.com/kinesis/

[4] Secure Cloud Storage with AWS S3 - https://aws.amazon.com/s3/

[5] Best Practices for Fraud Prevention - https://www.fraudpractice.com/

[6] Identity and Access Management (IAM) Best Practices - https://aws.amazon.com/iam/

[7] Implementing Serverless Architectures - https://aws.amazon.com/serverless/