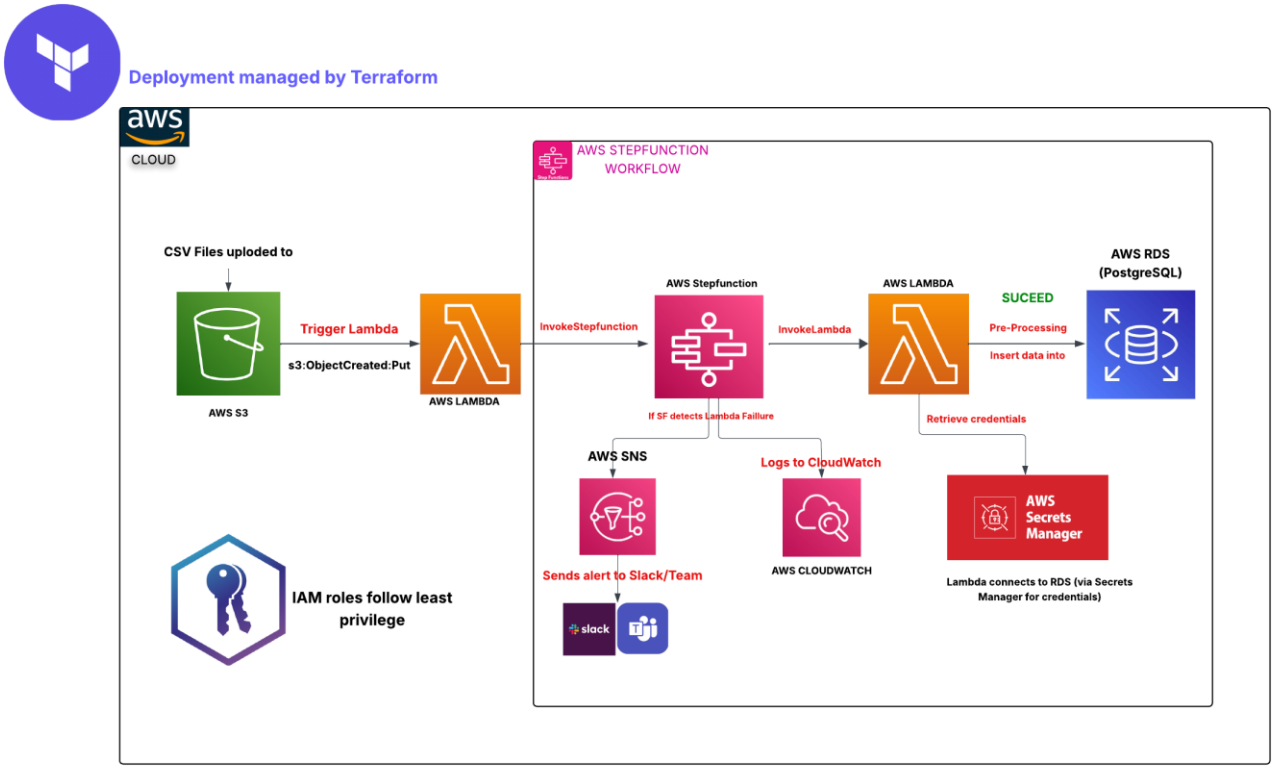
Deployment :

Proposed Architecture (Serverless & IaC) :

The proposed architecture is fully based on serverless AWS services and is managed using Terraform for Infrastructure as Code (IaC).



Architecture components :

- AWS S3 for file drops (trigger point)
- AWS Lambda to trigger a Step Function
- Step Function orchestrates the ETL flow
- A dedicated Lambda for pre-processing & DB insertion
- AWS RDS (PostgreSQL) as your transactional DB

- CloudWatch for logs/monitoring
- SNS + Slack/Teams for alerting
- IAM with least privilege
- Secrets Manager for secure credentials
- Terraform for infrastructure-as-code (IaC).

Explanation:

1. **File Upload:**

   CSV files containing transaction data are uploaded to an Amazon S3 bucket. An **s3:ObjectCreated:Put** event automatically triggers an AWS Lambda function. This Lambda's only responsibility is to start the AWS Step Functions workflow, which orchestrates the ETL process.

2. **Pre-Processing, Transformation & Load:**

   Once invoked, the Step Functions workflow executes a dedicated Lambda function responsible for the full **ETL logic**:
   - **Pre-Processing:** File format validation, column checks, and schema validation.
   - **Data Transformation:** Type casting, tax calculations, normalization.
   - **Data Load:** Inserts the cleaned data into an Amazon RDS (PostgreSQL)

   The Lambda function retrieves credentials securely from **AWS Secrets Manager**, ensuring that sensitive data (like DB passwords) is not hardcoded and is managed according to best practices.

3. **Error Handling & Observability:**

   If any error occurs during the Step Function execution:
   - **AWS SNS** is triggered to send an alert
   - Notifications are sent to a Slack or Microsoft Teams channel via webhook

   CloudWatch collects logs for both successful and failed executions, providing full visibility for monitoring and troubleshooting.

   .
4. **Security & Best Practices:**

- All components use **least-privilege IAM** roles for secure access control.
- Lambda functions retrieve **RDS credentials** from **AWS Secrets Manager**, avoiding hardcoded secrets.
- The entire infrastructure is managed using **Terraform**, enabling reproducibility, auditability, and consistent environment management across stages.

Why This Architecture?

This serverless architecture is simple, efficient, and easy to maintain. A lightweight Lambda function triggers a **Step Functions** workflow, which centralizes the processing (validation, transformation, and insertion into RDS) through a **single dedicated Lambda**.

Scalability & Evolution:

Step Functions make it easy to add **parallel steps** as the workflow evolves. If the data volume increases, the architecture can be adapted to a **batch processing** approach using **AWS Glue** for better performance and scalability.

NB: I didn't mention that the second Lambda function needs to connect securely to the RDS within a VPC and a private subnet, simply to keep the schema/architecture clear.