Fenix Alliance

Salesforce Sync Lambda Functions

Project Documentation
Client: Italcol, Santa Reyes & Casablanca
Version: 1.0
Date: May 8, 2025

---

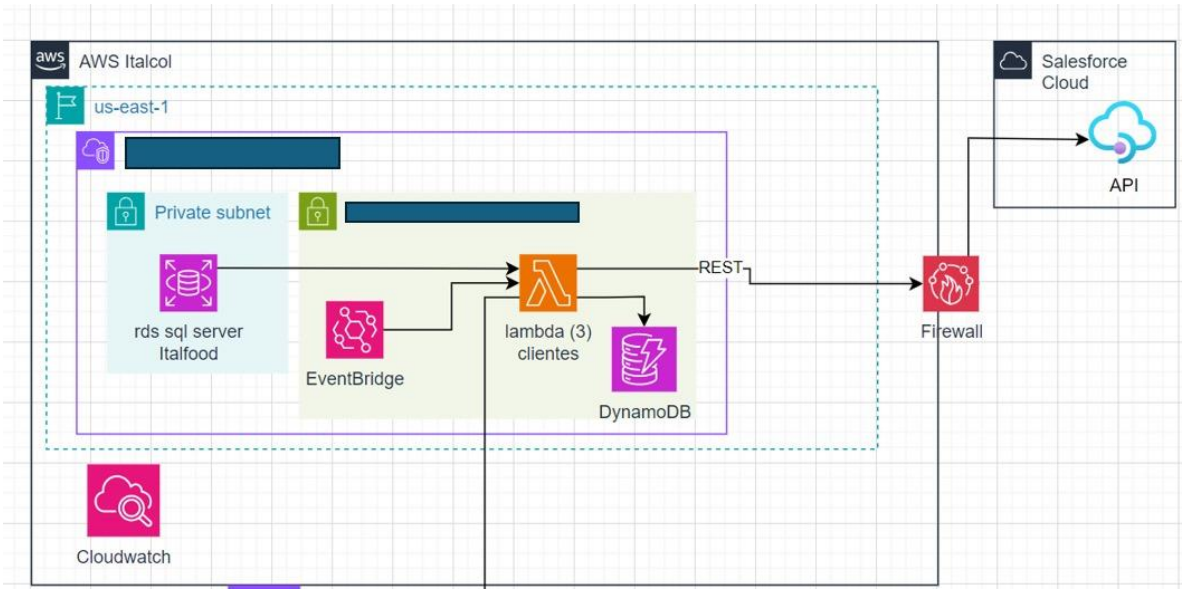
1. Executive Summary

This document describes the design, configuration, and operation of a set of AWS Lambda functions developed to synchronize business data from SQL Server into Salesforce. The solution ensures accurate, daily updates of Accounts, Retail Stores, and Agreements, aligning internal data sources with Salesforce for business continuity and operational efficiency.

Key Features:

- Automatic daily synchronization at 5:00 AM UTC.
- Seamless integration between SQL Server and Salesforce via REST API.
- Modular, configurable system with minimal maintenance overhead.
- Uses AWS-managed infrastructure for scalability and reliability.

Architecture Overview:



The Lambda suite includes:

- Terceros: synchronizes **Accounts.**
- Sucursales: synchronizes **Retail Stores.**
- Acuerdos: synchronizes **Agreements.**

---

2. Usage & Scheduling

Triggering

Each Lambda is triggered by **Amazon EventBridge** on a daily schedule.

- **Cron Expression:** cron(0 5 * * ? *)
  (Executes daily at 5:00 AM UTC)

Input

The Lambdas require an input event with a selected company:

```
{
  "selected_company": "ITALCOL"
}
```

Accepted values: `ITALCOL`, `CASABLANCA`, `SANTA_REYES`.

Execution Flow

1. Triggered by EventBridge.
2. Loads configuration from config.json.
3. Connects to the correct SQL database based on selected_company.
4. Executes a SQL view to retrieve data.
5. Maps SQL records into Salesforce-compatible JSON.
6. Authenticates with Salesforce using OAuth2 password grant.
7. For each record:
    - Checks if it already exists in Salesforce (via external_id).
    - If it exists, performs a **PATCH**.
    - If it doesn't, performs a **POST**.
    - Tracks or updates the mapping in DynamoDB.

---

3. Deployment Overview

This solution is deployed using AWS Lambda with a **manual ZIP packaging process**. There is no CI/CD pipeline, as the system is not expected to change frequently.

Deployment Steps (Summary)

1. Package the Lambda code and dependencies into a ZIP file.
2. Upload via AWS Console or AWS CLI.
3. Set environment variables manually.
4. Configure EventBridge rule for scheduled execution.

Note: All secrets and credentials are embedded via environment variables or the bundled config.json.

---

4. Configuration Details

`config.json` Sample

```
{
  "VIEW": "SELECT * FROM sucursales_view WHERE ...",
  "AUTH_URL": "https://login.salesforce.com/services/oauth2/token",
  "AUTH_PARAMS": {
    "grant_type": "password",
    "client_id": "...",
    "client_secret": "...",
    "username": "...",
    "password": "..."
  },
  "API_URI": "/services/data/v58.0/sobjects/ti_sucursal__c/"
}
```

Environment Variables

Variable	Description
`DB_HOST`	Default SQL Server host
`DB_HOST_ITALCOL`	Alternate host for ITALCOL
`DB_NAME`	Default DB name
`DB_NAME1/2`	Additional DBs for specific companies
`DB_USER`	SQL Server username
`DB_PASSWORD`	SQL Server password

5. Data Flow & Logic

End-to-End Flow

1. Load config.json.
2. Connect to SQL Server and run the view.
3. Transform rows into Salesforce-compliant payloads.
4. Authenticate with Salesforce.
5. For each record:
    - Retrieve external ID from DynamoDB.
    - Insert (POST) or update (PATCH) record in Salesforce.
    - Track the association in DynamoDB.

Lambda Responsibilities

Lambda	Description	SQL View	Salesforce Object
`terceros`	Syncs customer accounts	terceros_view	Account
`sucursales`	Syncs retail store data	sucursales_view	RetailStores
`acuerdos`	Syncs agreement records	acuerdos_view	ti_acuerdo__c

---

6. Monitoring & Logs

The system uses **Amazon CloudWatch** for logging and monitoring. Each Lambda emits structured logs to help with debugging and tracking synchronization status.

Logged Information Includes:

- Start and end time
- Number of records processed
- Successes and failures
- Error messages or stack traces

Recommendation: Set up CloudWatch Alarms to alert Lambda errors or execution failures.

---

7. Maintenance & Handover

When to Update the Configuration

- SQL views are modified or renamed.
- A new company added to the system.
- Salesforce API credentials are rotated.

Maintenance Checklist

Task	Frequency	Notes
Rotate Salesforce credentials	Every 90 days	As per Salesforce security policy
Review config and views	As needed	Ensure SQL and API are aligned
Monitor CloudWatch logs	Weekly	Identify unusual failures early

---

8. Support Contact

For issues, updates, or maintenance requests, please contact:

- **Email:** support@fenix-alliance.com
- **Lead Developer:** diego.ballesteros@fenix-alliance.com

9. Appendix

Project Directory Structure

```
├── main.py
├── config/
│   └── config_loader.py
├── services/
│   ├── dynamo_service.py
│   ├── database_service.py
│   ├── mapper_service.py
│   └── salesforce_service.py
└── constants/
    └── allowed_companies.py
```

Module Responsibilities

- `main.py`: Entry point and orchestrator.
- `config_loader.py`: Loads and validates config.
- `database_service.py`: Connects to SQL Server and fetches data.
- `mapper_service.py`: Converts SQL rows to Salesforce format.

- `salesforce_service.py`: Authenticates and sends data to Salesforce.
- `dynamo_service.py`: Manages ID mappings via DynamoDB.
- `allowed_companies.py`: Contains whitelisted company codes.