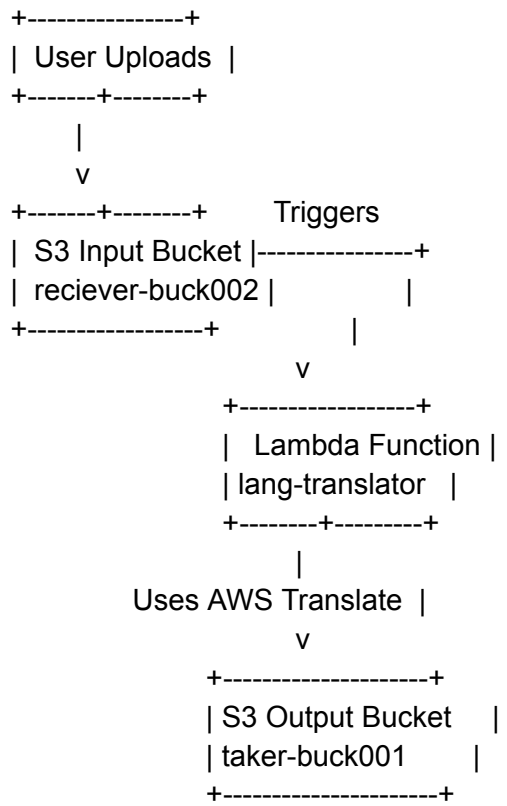
AWS Automated language translation pipeline

Project Overview

*This project implements a **serverless language translation pipeline** using AWS. The goal is to **automatically translate uploaded text files** from an input S3 bucket and **store the translated output** in a separate output bucket using Amazon Translate and AWS Lambda.*

1.

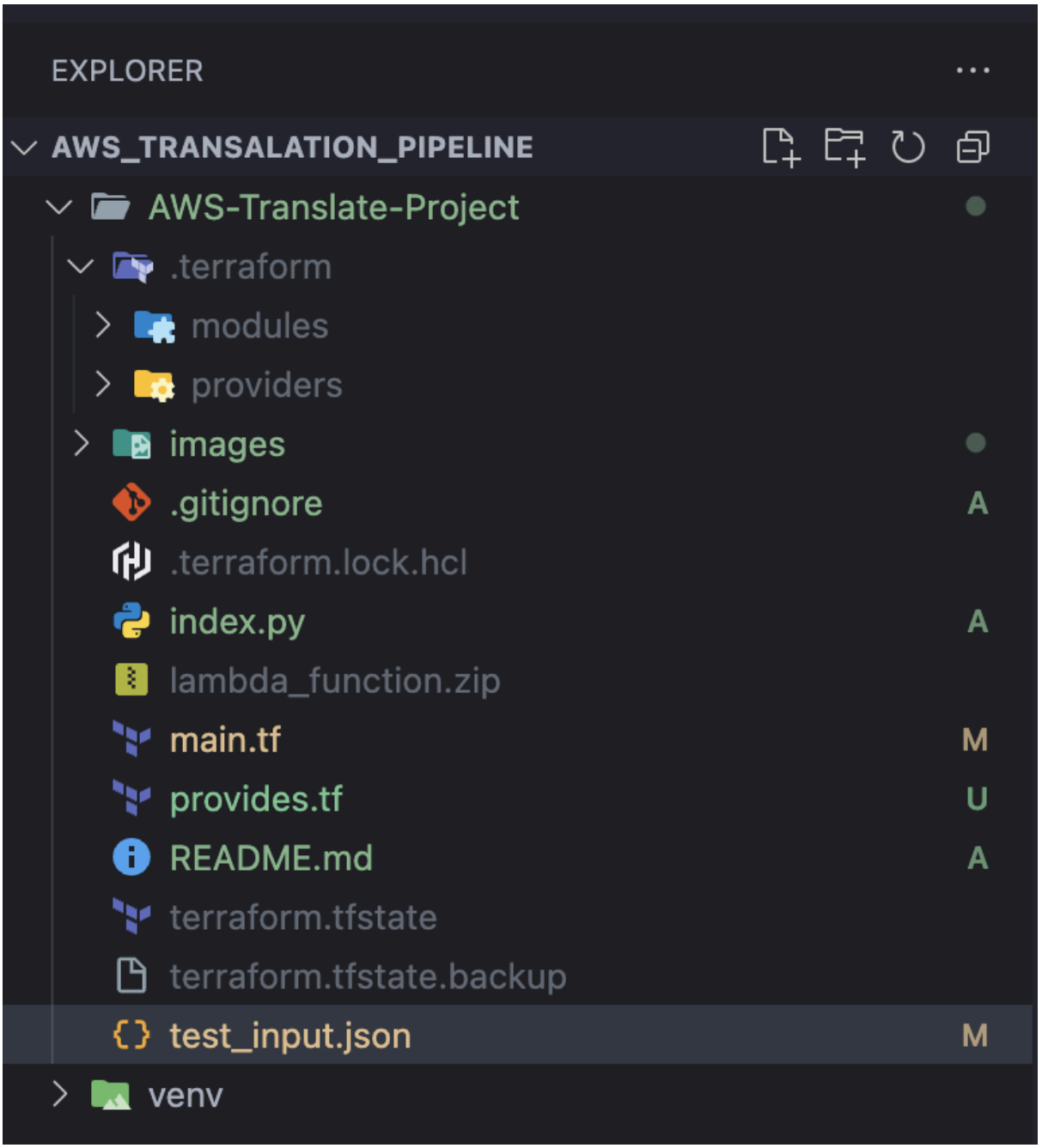
Workflow diagram



Project Structure

```
translation-pipeline/
│
├── terraform/
│   ├── main.tf                # All infrastructure defined here
│   ├── providers.tf
│
```

```
├── lambda/
│   ├── index.py                # Lambda function logic
│   └── lambda_function.zip     # Zipped package for deployment
│
├── test/
│   └── test_input.json         # Sample input file for testing
│
└── README.md                   # Project documentation
```



Tools and resources

Component	Service/Tool	Purpose
Compute	AWS Lambda	Serverless backend for translation
File Storage	Amazon s3	Input and Output buckets
IAM	AWS IAM Roles/Policies	Secure access between services
Infrastructure	Terraform	IAC to deploy all resources
Monitoring	Cloudwatch	Lambda execution logging
Translation API	Amazon Translate	Actual Text translation

IAM & Permissions

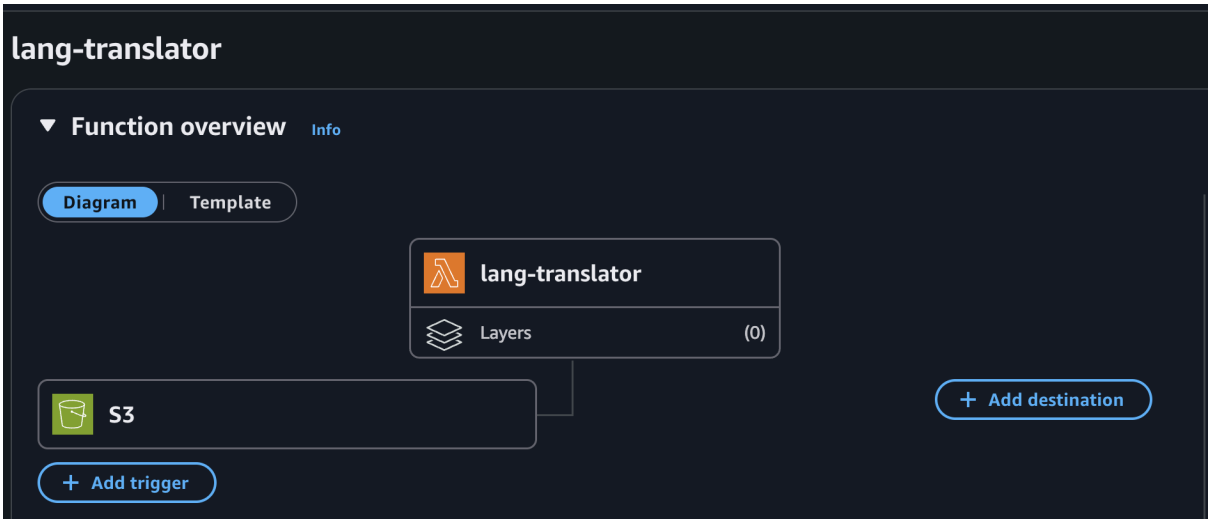
The Lambda function has an IAM role that allows:

- translate: TranslateText
- s3:GetObject, s3:ListBucket on input bucket
- s3:PutObject on output bucket
- logs:* for writing to CloudWatch
-

Event Trigger Flow

2. A JSON file is uploaded to receiver-buck002.
3. An **S3 event notification** triggers the lang-translator Lambda.

4. Lambda reads the file and extracts translation fields.
5. Uses Amazon Translate to translate text.
6. Stores the result in taker-buck001.



Sample Lambda Input Format

AWS-Translate-Project > {} test_input.json > ...

```
[
  {
    "Text": "Hello, Welcome Back?",
    "SourceLanguageCode": "en",
    "TargetLanguageCode": "fr"
  },
  {
    "Text": "I am part of IAM Masters.",
    "SourceLanguageCode": "en",
    "TargetLanguageCode": "es"
  }
]
```

Sample Output File

test_input-translated.json 〉 No Selection


```
{"translations": [{"original": "Hello, Welcome Back?", "translated": "Bonjour, bon retour\u00a0?", "from": "en", "to": "fr"}, {"original": "I am part of IAM Masters.", "translated": "Formo parte de IAM Masters.", "from": "en", "to": "es"}]}
```


How to Deploy

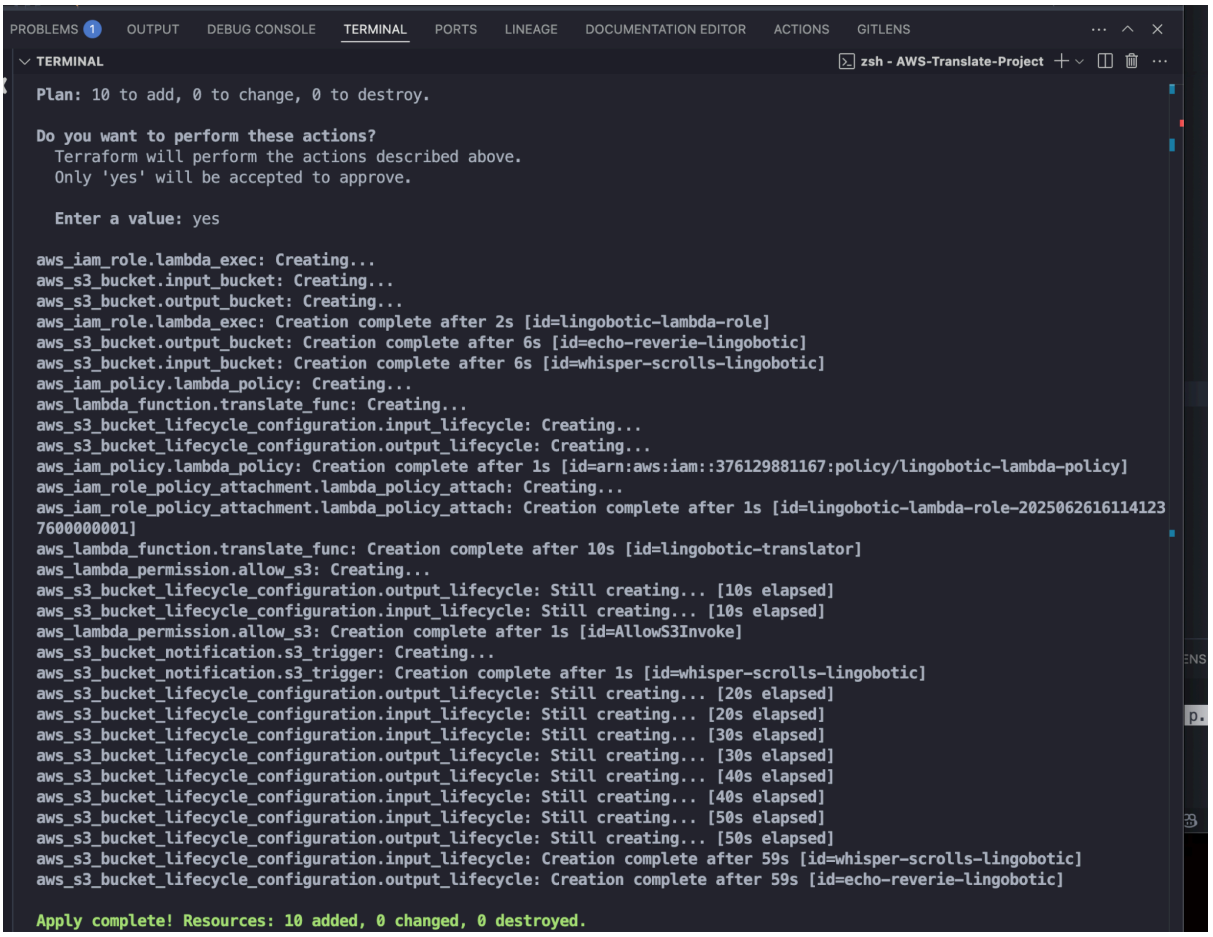
1. **Install Terraform**

```
# ---------------------------
# 1. S3 BUCKETS
# ---------------------------

resource "aws_s3_bucket" "input_bucket" {
  bucket = "reciever-buck002"
  force_destroy = true
}

resource "aws_s3_bucket" "output_bucket" {
  bucket = "taker-buck001"
  force_destroy = true
}

# Lifecycle for input bucket
resource "aws_s3_bucket_lifecycle_configuration" "input_lifecycle" {
  bucket = aws_s3_bucket.input_bucket.id

  rule {
    id     = "expire-inputs"
    status = "Enabled"

    filter {
      prefix = ""
    }

    expiration {
      days = 30
    }
  }
}

# Lifecycle for output bucket
resource "aws_s3_bucket_lifecycle_configuration" "output_lifecycle" {
  bucket = aws_s3_bucket.output_bucket.id
```

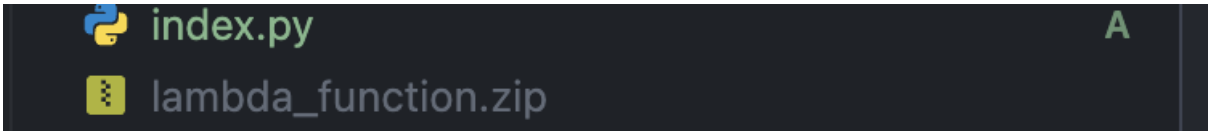
terraform init

Terraform plan

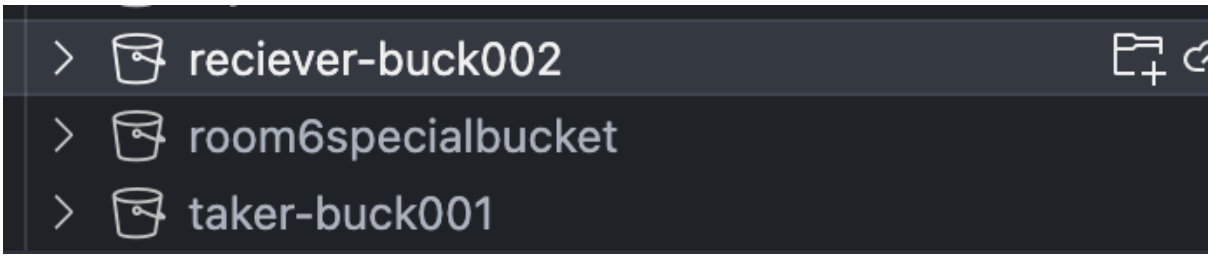
4. **Deploy to Lambda** via Terraform or manually.



zip lambda_function.zip index.py



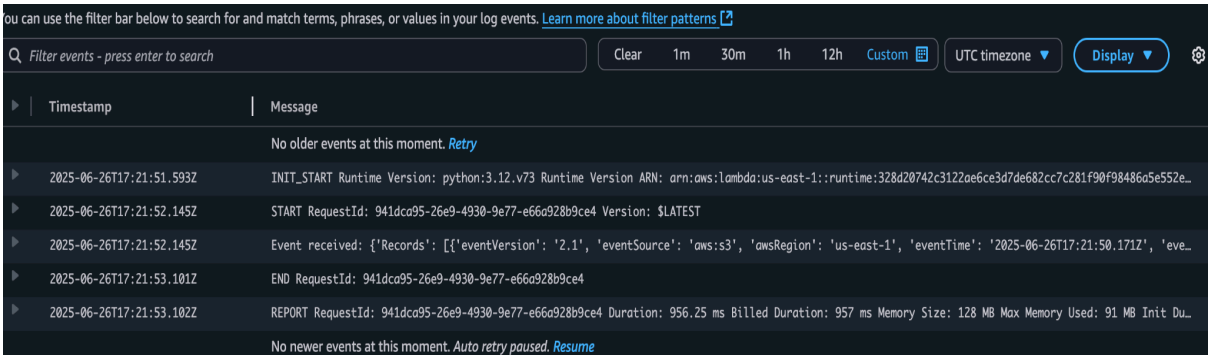
5. Check if resources have been created, eg, buckets

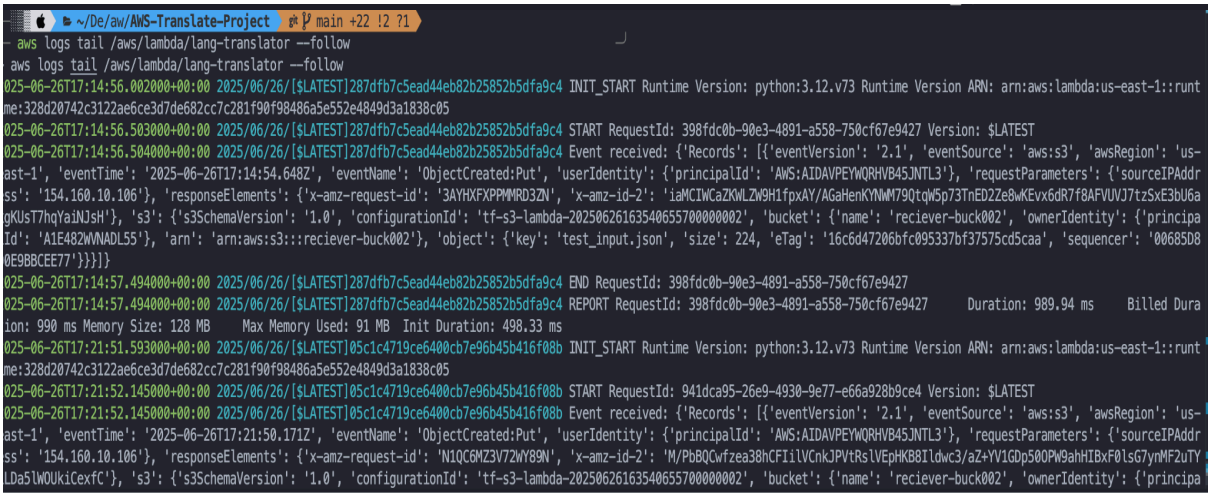


6. **Upload test file**:
   aws s3 cp ../test/test_input.json s3://reciever-buck002/

Monitoring

- Use CloudWatch Logs:

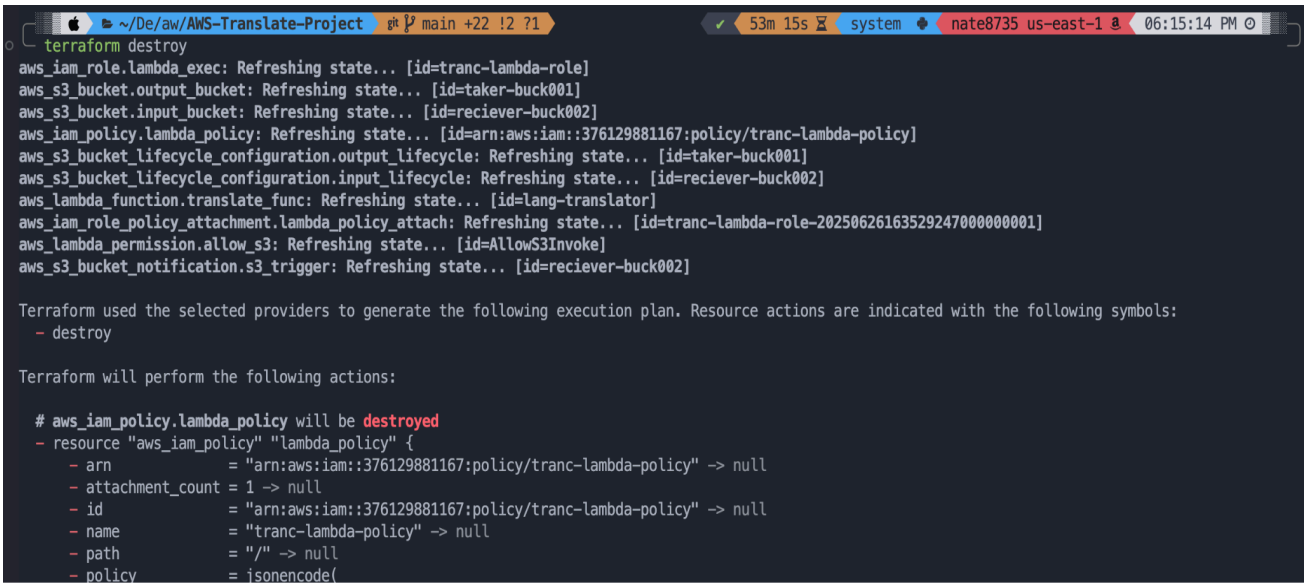




Future Improvements

- Add SNS notifications on successful translation
- Add DynamoDB to store translation history
- Add a front-end upload interface
- Include language detection (if SourceLanguageCode is missing)
- Validate file content format before processing

Cleanup terraform destroy



IAM Masters

Nathaniel Sackey

Chrysolite Yeboah

Emmanuel Osei

Kelvin NeeQuaye

Eric Datsa

Mariam Zakiyu

GitHub: Github repo