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.*



Workflow diagram

+----------------+

| User Uploads |

+-------+--------+

|

v

+-------+--------+ Triggers

| S3 Input Bucket |----------------+

| reciever-buck002 | |

+------------------+ |

v

+------------------+

| Lambda Function |

| lang-translator |

+--------+---------+

|

Uses AWS Translate |

v

+---------------------+

| S3 Output Bucket |

| taker-buck001 |

+----------------------+

**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

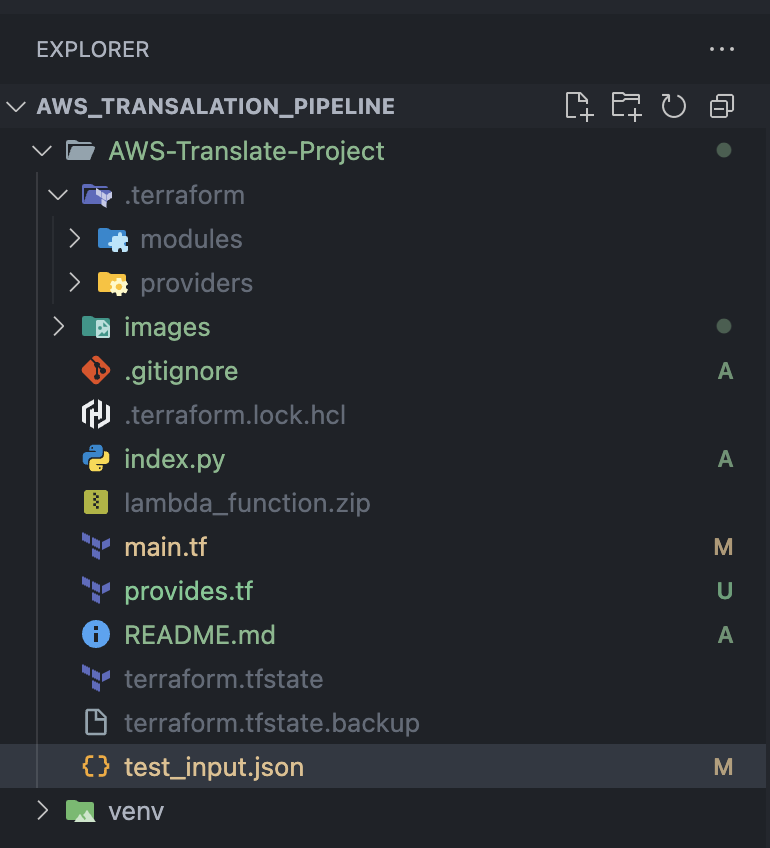
│

├── 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 |

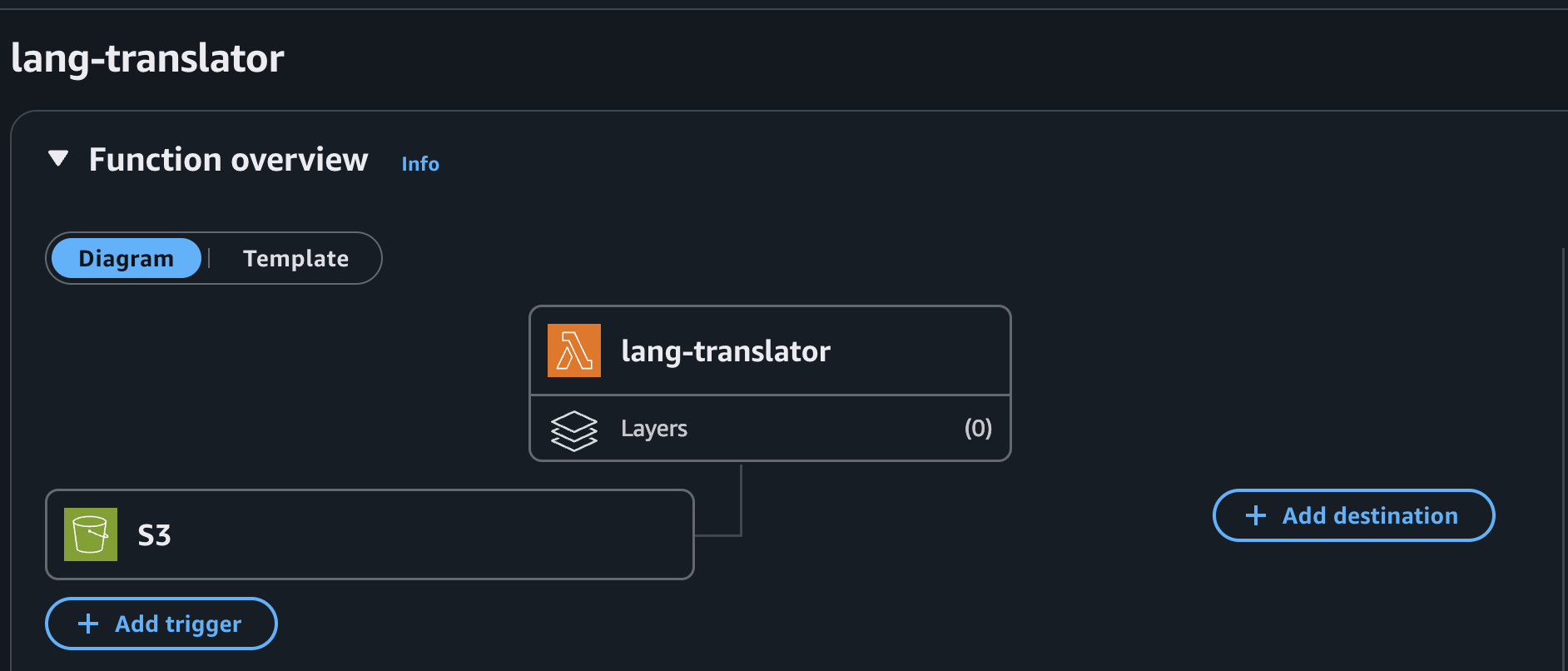
### **IAM & Permissions**

The Lambda function has an IAM role that allows:

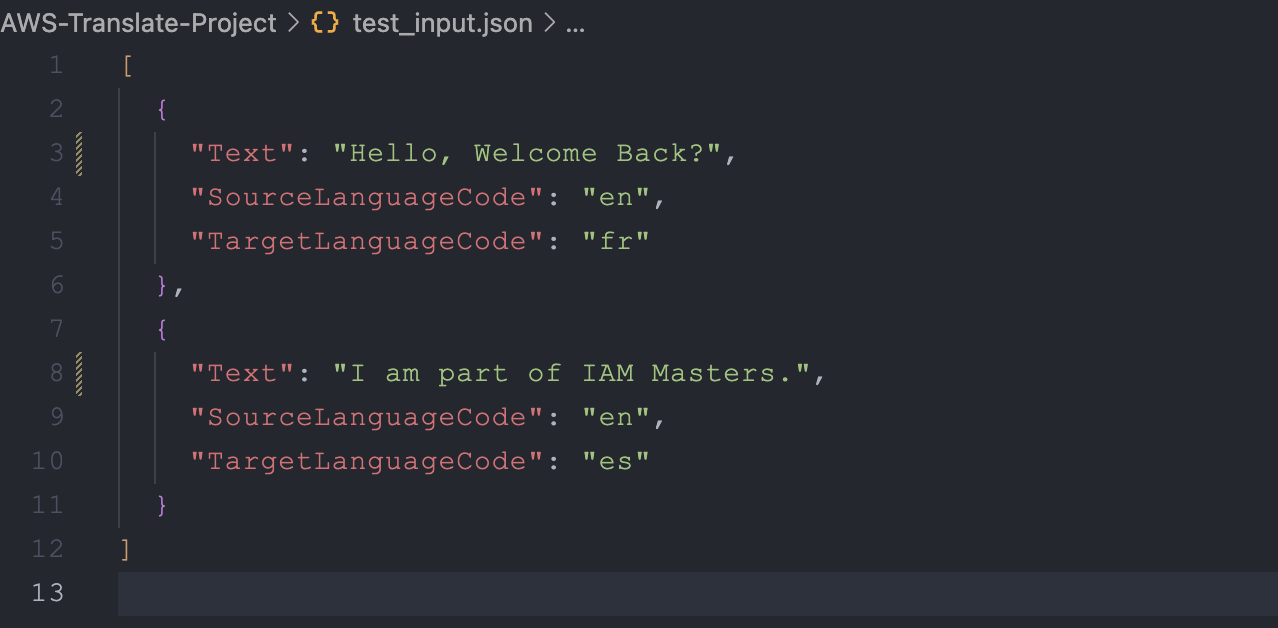
* translate: TranslateText
* s3:GetObject, s3:ListBucket on input bucket
* s3:PutObject on output bucket
* logs:\* for writing to CloudWatch

### **Event Trigger Flow**

1. A JSON file is uploaded to receiver-buck002.
2. An **S3 event notification** triggers the lang-translator Lambda.
3. Lambda reads the file and extracts translation fields.
4. Uses Amazon Translate to translate text.
5. Stores the result in taker-buck001.



### **Sample Lambda Input Format**



### **Sample Output File**



### **How to Deploy**

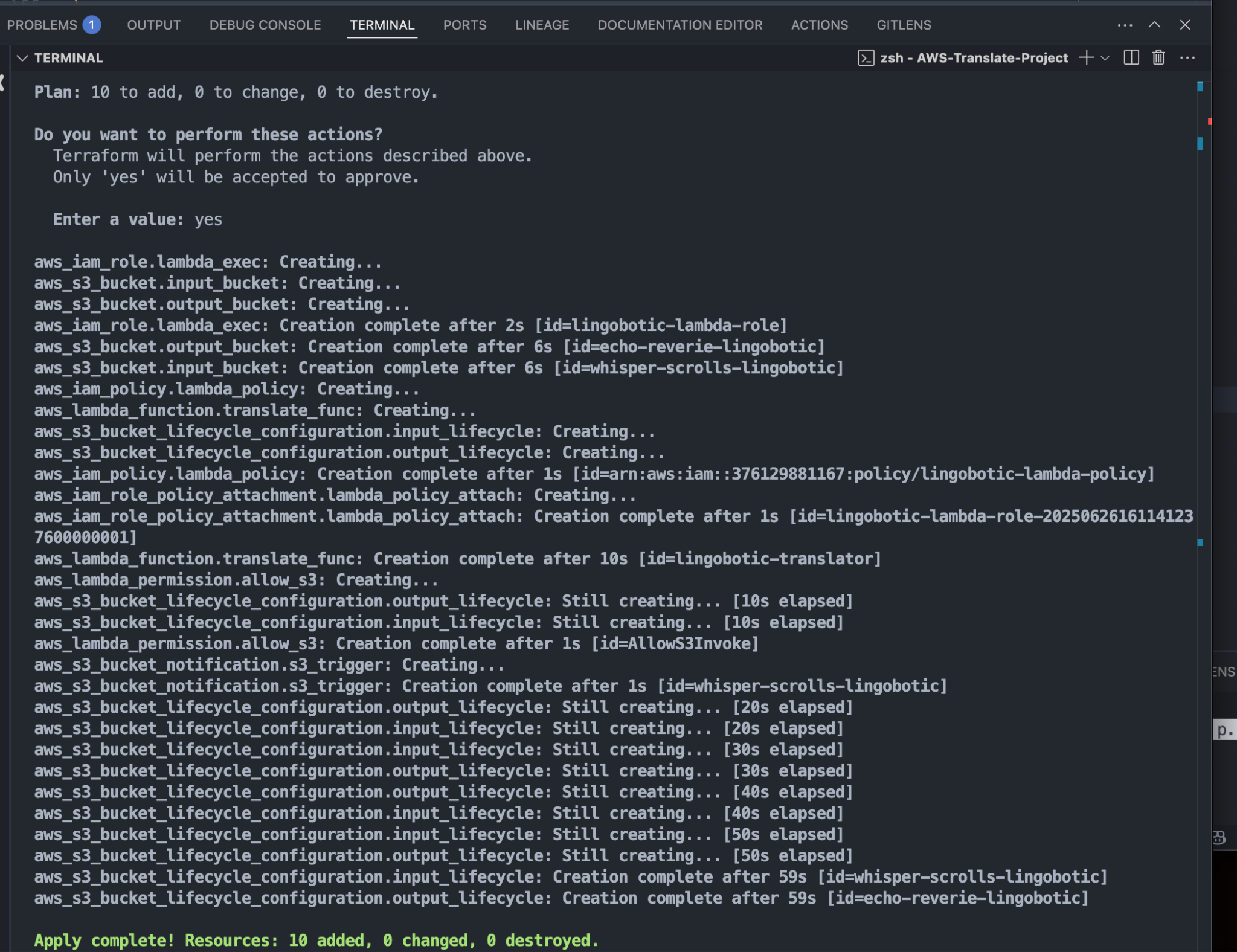
1. **Install Terraform**



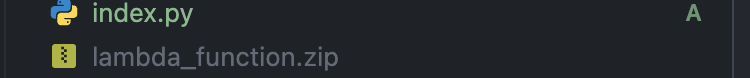
terraform init

Terraform plan

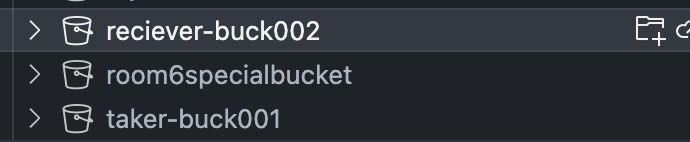
1. **Deploy to Lambda** via Terraform or manually.



zip lambda\_function.zip [index.py](http://index.py)



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

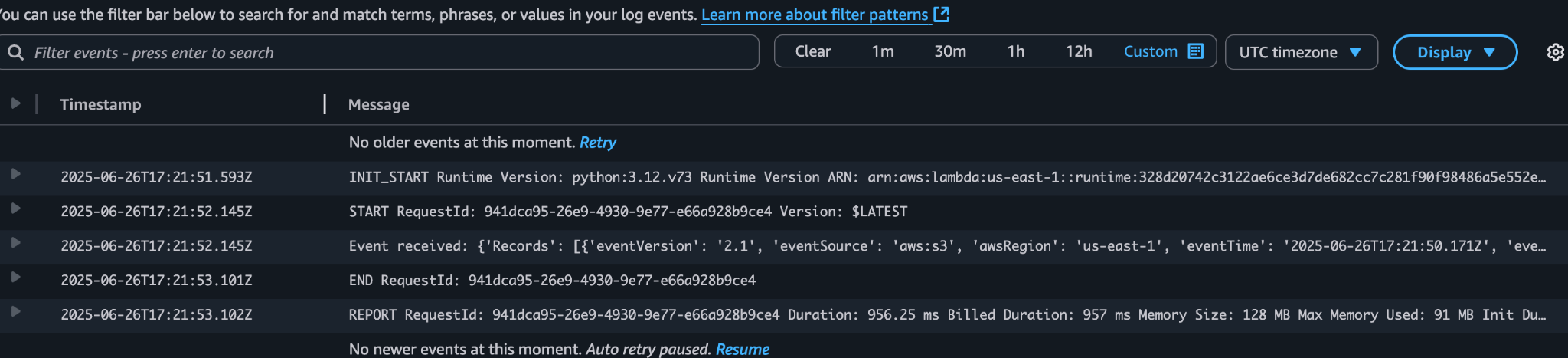


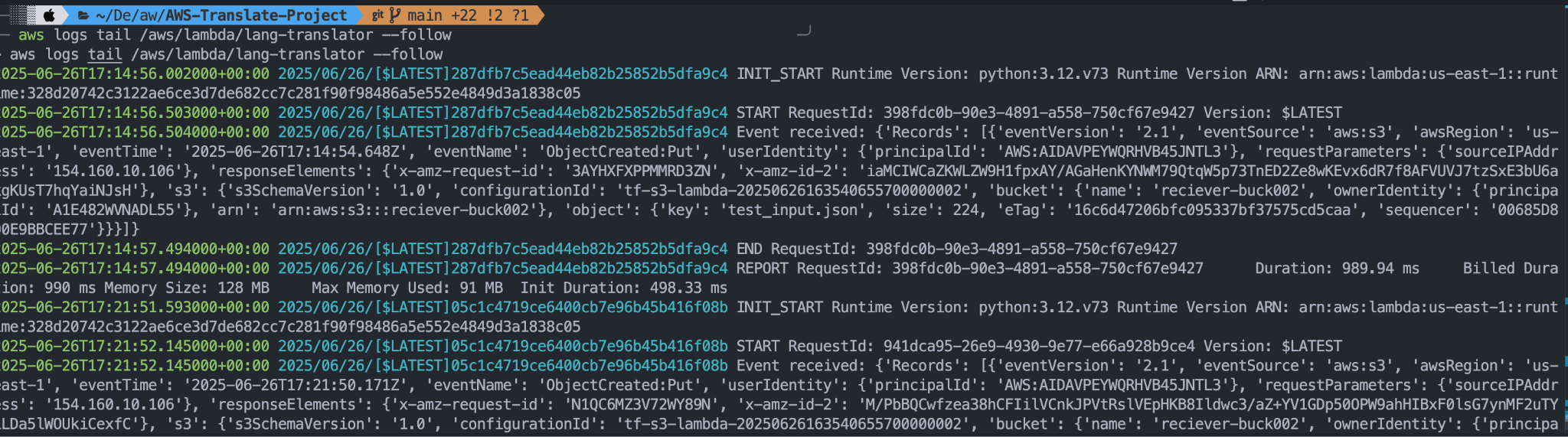
1. **Upload test file**:  
   aws s3 cp ../test/test\_input.json s3://reciever-buck002/

### 

### **Monitoring**

* Use CloudWatch Logs:



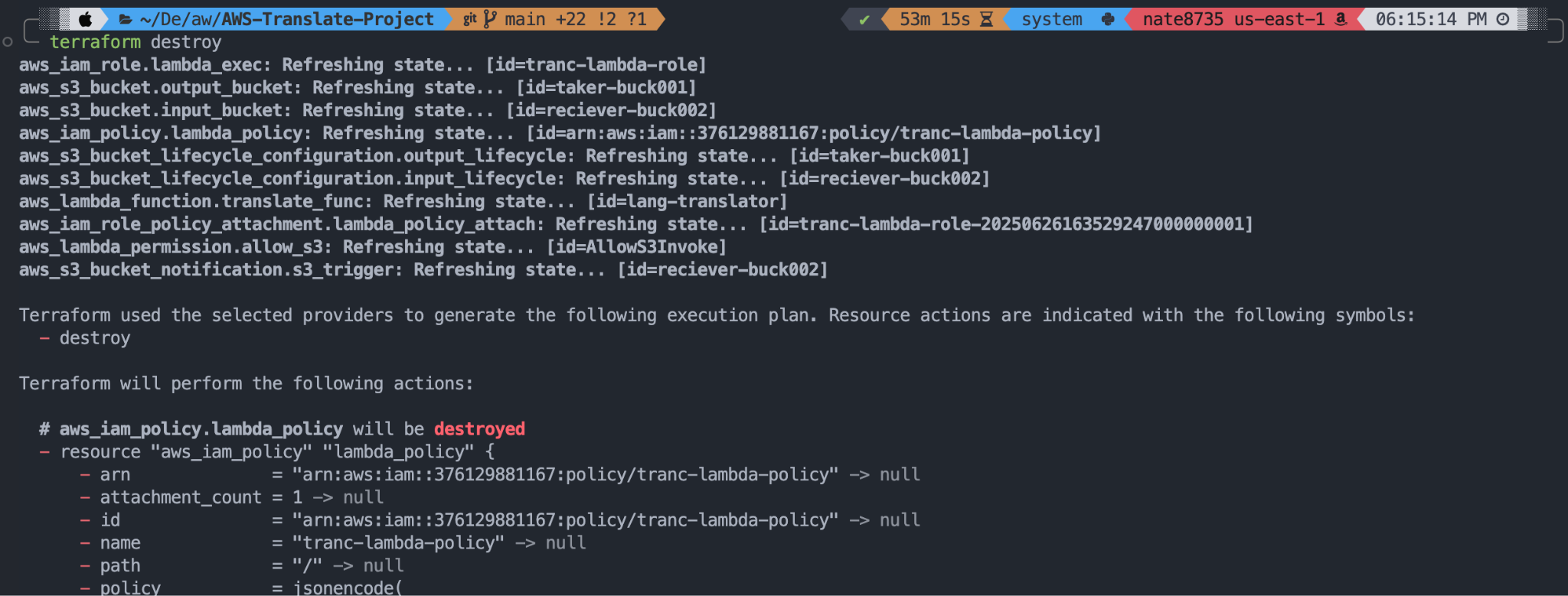


### 

### **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](https://github.com/bignate8735/transalation_pipeline)