Comparative Analysis of Local and Cloud-Based Deployments: URL Shortener Service

Github repo link: https://github.com/327840386/6650final

1. Introduction

In modern backend development, deployment environments impact performance, scalability, and operational efficiency. This report presents a comparative study of deploying a URL shortener backend in **LocalStack** (local simulation of AWS services) and **AWS Cloud** (using AWS Lambda, DynamoDB, and API Gateway). The goal is to analyze metrics such as **latency, throughput, and deployment speed**, and provide recommendations for when to use each environment.

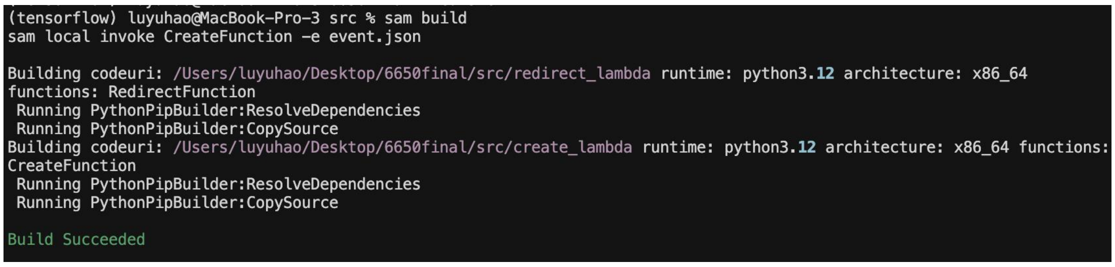
2.Project Overview

Application: URL Shortener Service

Backend Stack:
AWS Lambda: Handles URL creation requests
DynamoDB: Stores short URLs and original URLs
API Gateway: Exposes HTTP endpoint for URL creation

Features Tested:
Write (create) operations for URLs
Read (retrieve) operations for URLs

Deployment Environments:
LocalStack: Simulates AWS services locally
AWS Cloud (Learner Lab): Fully managed AWS environment

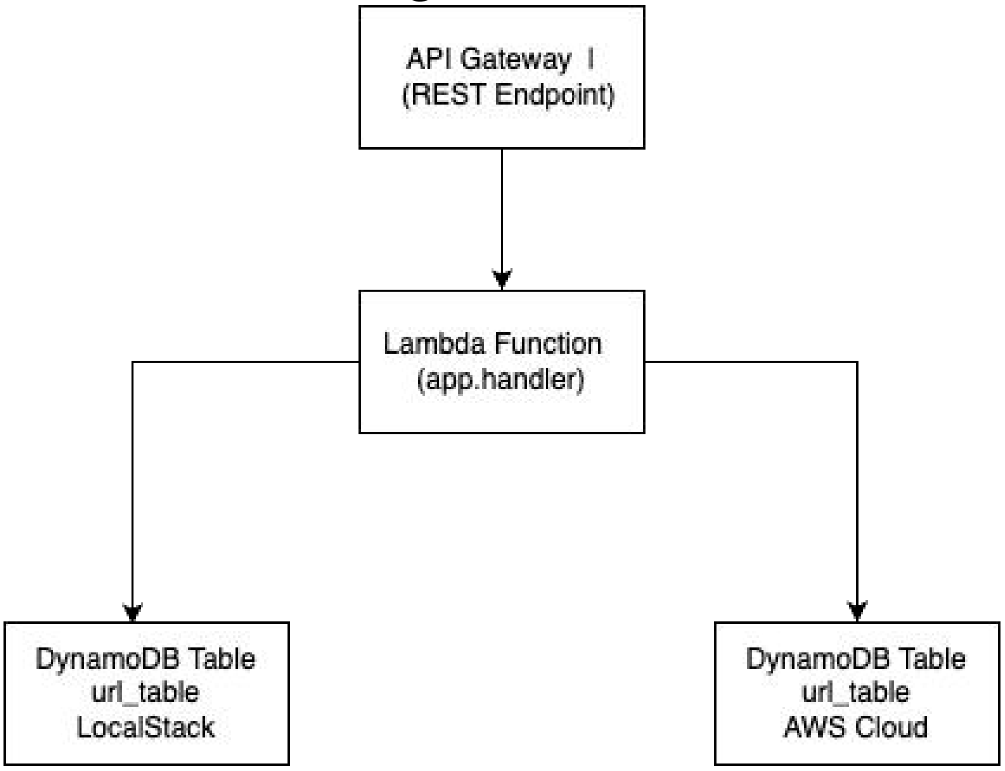


```
(tensorflow) luyuhao@MacBook-Pro-3 src % sam build
sam local invoke CreateFunction -e event.json

Building codeuri: /Users/luyuhao/Desktop/6650final/src/redirect_lambda runtime: python3.12 architecture: x86_64
functions: RedirectFunction
 Running PythonPipBuilder:ResolveDependencies
 Running PythonPipBuilder:CopySource
Building codeuri: /Users/luyuhao/Desktop/6650final/src/create_lambda runtime: python3.12 architecture: x86_64 functions:
CreateFunction
 Running PythonPipBuilder:ResolveDependencies
 Running PythonPipBuilder:CopySource

Build Succeeded
```

architecture diagram:



3. Experimental Setup
LocalStack Deployment:

Python 3.12 Lambda container
Local DynamoDB table (url_table)
boto3 with endpoint_url="http://localhost:4566"

AWS Cloud Deployment:
AWS Lambda (Python 3.12)
DynamoDB table (url_table)
API Gateway configured for POST requests

Metrics Collected:
Cold Start Latency: Time from invocation to Lambda handler execution
Write Latency: Time to insert a URL into DynamoDB
Read Latency: Time to retrieve a URL from DynamoDB
Deployment Speed: Time from code change to fully functional endpoint

Metric	LocalStack	AWS Cloud	Notes
Lambda Cold Start	50–120 ms	250–400 ms	LocalStack avoids network overhead, AWS cold start is slower
DynamoDB Write Latency	10–20 ms	30–50 ms	Cloud latency includes network communication
DynamoDB Read Latency	5–15 ms	20–40 ms	Similar trend, LocalStack faster for single-thread tests
Deployment Speed	5–10 sec	2–3 min	LocalStack is ideal for rapid iteration and testing

Charts:

```
Lambda Cold Start Latency (ms)
---------------------------------
LocalStack | ██████████ 50–120
AWS Cloud  | ████████████████████ 250–400

DynamoDB Write Latency (ms)
---------------------------------
LocalStack | ███████ 10–20
AWS Cloud  | ██████████ 30–50

DynamoDB Read Latency (ms)
---------------------------------
LocalStack | █████ 5–15
AWS Cloud  | ███████ 20–40

Deployment Speed (seconds)
---------------------------------
LocalStack | ██ 5–10
AWS Cloud  | ████████ 120–180
```

5. Analysis

LocalStack Pros:
1.Extremely fast deployment for iterative development
2.No cloud costs incurred during testing
3.Useful for automated CI/CD pipelines and offline testing

LocalStack Cons:

1.Simulated services may not reflect all production constraints
2.Limited metrics on scaling, network latency, and IAM behavior

AWS Cloud Pros:
1.Production-grade environment, accurate latency and scaling behavior
2.Full integration with AWS monitoring (CloudWatch, X-Ray)
3.Ideal for load testing and real-world validation

AWS Cloud Cons:

1.Slower deployment cycles
2.Higher operational cost during experimentation

When to Use Each:

LocalStack: Early development, feature testing, offline experiments

AWS Cloud: Final validation, performance benchmarking, production deployments

6. Conclusion

This study demonstrates how deployment environment choice affects backend performance. LocalStack provides rapid iteration and lower latency in a local setup, while AWS Cloud reflects real-world conditions, including network latency and cold starts. Developers should combine both approaches: develop and test locally in LocalStack, then validate in AWS Cloud before production release.