**Deploying the Java Postfix Evaluator with AWS Amplify (Java Backend & JavaScript Frontend)**

This guide will walk you through setting up the provided Java code in AWS using Amplify for the frontend and AWS Lambda for the backend. The setup consists of:

1. Creating a Java-based AWS Lambda function to evaluate postfix expressions.
2. Exposing the function through AWS API Gateway.
3. Setting up an AWS Amplify JavaScript frontend to interact with the API.

**Step 1: Set Up AWS Lambda for Java Backend**

**1.1 Create a New AWS Lambda Function**

1. Log in to your **AWS Management Console**.
2. Navigate to **AWS Lambda** → **Create function**.
3. Choose **"Author from scratch"**.
4. Function name: PostfixEvaluatorLambda
5. Runtime: **Java 17**
6. Choose an execution role: Select **"Create a new role with basic Lambda permissions"**.
7. Click **"Create function"**.

**1.2 Write the Java Code for AWS Lambda**

Modify the Java code to be compatible with AWS Lambda. Replace System.out.println statements with proper API responses.

**Lambda Function Code (PostfixEvaluator.java)**

package com.example;

import com.amazonaws.services.lambda.runtime.Context;

import com.amazonaws.services.lambda.runtime.RequestHandler;

import java.util.Stack;

import java.util.Map;

public class PostfixEvaluator implements RequestHandler<Map<String, String>, String> {

@Override

public String handleRequest(Map<String, String> input, Context context) {

String expression = input.get("expression");

if (expression == null || expression.isEmpty()) {

return "Invalid input: Expression is required";

}

return evaluatePostfix(expression);

}

static String evaluatePostfix(String expression) {

Stack<Integer> stack = new Stack<>();

char[] array = expression.toCharArray();

for (char c : array) {

if (Character.isDigit(c)) {

stack.push(c - '0'); // Convert char to int

} else {

if (stack.size() < 2) {

return "Invalid postfix expression";

}

int operand2 = stack.pop();

int operand1 = stack.pop();

int result = evaluate(operand1, operand2, c);

stack.push(result);

}

}

return stack.isEmpty() ? "Invalid expression" : String.valueOf(stack.pop());

}

static int evaluate(int operand1, int operand2, char operator) {

switch (operator) {

case '+': return operand1 + operand2;

case '-': return operand1 - operand2;

case '\*': return operand1 \* operand2;

case '/': return operand2 == 0 ? 0 : operand1 / operand2;

default: return 0;

}

}

}

**1.3 Package and Deploy the Lambda Function**

1. **Compile the Java Code**
   * Open a terminal and navigate to your project directory.
   * Use Maven or Gradle to package the project as a JAR file.

Using Maven:

mvn clean package

The output JAR file will be located in the target directory.

1. **Upload the JAR to AWS Lambda**
   * Go to the Lambda function in the AWS console.
   * Under **"Code"**, click **"Upload from" → ".zip or .jar file"**.
   * Select your target/\*.jar file and upload it.

**Step 2: Configure API Gateway to Expose Lambda Function**

1. Go to **AWS API Gateway** → **Create API**.
2. Select **"HTTP API"** → **Build**.
3. Click **"Add Integration"** → Select **AWS Lambda** → Choose your PostfixEvaluatorLambda function.
4. Set up a **POST method** with the route /evaluate.
5. Deploy the API and note down the **API endpoint URL** (e.g., https://xyz123.execute-api.us-east-1.amazonaws.com/evaluate).

**Step 3: Set Up AWS Amplify for the JavaScript Frontend**

**3.1 Install AWS Amplify CLI**

If you haven't installed AWS Amplify, install it using:

npm install -g @aws-amplify/cli

amplify configure

Follow the on-screen instructions to set up Amplify with your AWS account.

**3.2 Create an Amplify Project**

npx create-react-app postfix-evaluator

cd postfix-evaluator

amplify init

* Choose default settings and connect it to your AWS account.

**3.3 Add API Integration**

amplify add api

* Choose **REST API**
* Select **Use an existing API**
* Enter the API Gateway URL from Step 2

Run:

amplify push

**Step 4: Create the Frontend UI**

**4.1 Install AWS Amplify in the React Project**

npm install aws-amplify @aws-amplify/api

**4.2 Configure AWS Amplify (src/aws-exports.js)**

AWS Amplify automatically generates this file during amplify push. Ensure it includes the API configuration.

**4.3 Create JavaScript Function to Call the API**

Create a new file src/api.js:

import { API } from 'aws-amplify';

export async function evaluateExpression(expression) {

try {

const response = await API.post('postfixEvaluatorApi', '/evaluate', {

body: { expression },

});

return response;

} catch (error) {

console.error("Error evaluating expression:", error);

return "Error evaluating expression";

}

}

**4.4 Create the HTML & React UI**

Replace src/App.js with the following:

import React, { useState } from 'react';

import { evaluateExpression } from './api';

function App() {

const [expression, setExpression] = useState('');

const [result, setResult] = useState('');

const handleEvaluate = async () => {

const response = await evaluateExpression(expression);

setResult(response);

};

return (

<div style={{ textAlign: 'center', marginTop: '50px' }}>

<h2>Postfix Expression Evaluator</h2>

<input

type="text"

value={expression}

onChange={(e) => setExpression(e.target.value)}

placeholder="Enter postfix expression"

/>

<button onClick={handleEvaluate}>Evaluate</button>

<h3>Result: {result}</h3>

</div>

);

}

export default App;

**4.5 Start the Frontend Application**

npm start

* This will launch a local web server at http://localhost:3000/.
* Enter a postfix expression (e.g., "22\*3+"), click **Evaluate**, and see the result from the Lambda function.

**Step 5: Deploy the React App with AWS Amplify**

amplify add hosting

* Select **"Amazon CloudFront and S3"**.

Deploy it:

amplify publish

This will generate a public URL to access your React application.

**Conclusion**

This setup successfully deploys:

1. A **Java AWS Lambda function** to evaluate postfix expressions.
2. An **AWS API Gateway** to expose the function.
3. A **React frontend with AWS Amplify** to interact with the API.
4. Hosting the frontend using **AWS Amplify Hosting**.

This ensures a **serverless, scalable, and cost-effective solution** to evaluate postfix expressions using AWS. 🚀