# AWS API Gateway & Lambda Integration Lab

## Overview

This document details a hands-on lab where I successfully deployed an AWS Lambda function and integrated it with Amazon API Gateway. The lab demonstrates how API Gateway acts as a front-end to trigger a Lambda function, returning dynamic responses.

## Key Features

- Created an AWS Lambda function using a provided blueprint.

- Configured API Gateway to serve as a REST API trigger for the Lambda function.

- Tested the Lambda function both within the console and via API Gateway.

- Verified logs in Amazon CloudWatch to analyze execution details.

## Technical Skills & Tools Used

- AWS Services: Lambda, API Gateway, CloudWatch Logs.

- Security: IAM roles and API Gateway security settings.

- Debugging & Monitoring: CloudWatch Log Streams for execution insights.

## Implementation Steps

### Step 1: Create and Deploy AWS Lambda

• Created a new Lambda function using a pre-configured blueprint.

• Configured the function with:

- Node.js runtime

- Execution role: Assigned IAM permissions for API Gateway integration.

### Step 2: Configure API Gateway

• Created a REST API in API Gateway.

• Defined an endpoint to trigger the Lambda function.

• Set security permissions to allow access.

### Step 3: Test the API Gateway Trigger

• Deployed the API Gateway.

• Retrieved the API endpoint URL and tested the response in a web browser.

### Step 4: Debugging & Monitoring with CloudWatch

• Navigated to CloudWatch Logs under the Monitor tab in the Lambda console.

• Selected View CloudWatch Logs to analyze function executions.

• Opened Log Streams to inspect details, including:

- Execution time.

- Any error messages.

- Lambda function responses.

## Lambda Function Code (Node.js)

exports.handler = async (event) => {  
 const json\_questions = [  
 { question: "Can I access the infrastructure that AWS Lambda runs on?", answer: "No. AWS Lambda operates the compute infrastructure on your behalf, allowing it to perform health checks, apply security patches, and do other routine maintenance." },  
 { question: "What is the AWS Lambda execution environment?", answer: "AWS Lambda provides the Amazon Linux build of OpenJDK 11." }  
 ];  
  
 const rand = Math.floor(Math.random() \* json\_questions.length);  
 const response = {  
 statusCode: 200,  
 body: JSON.stringify(json\_questions[rand]),  
 };  
  
 console.log(response);  
 return response;  
};

## Challenges & Solutions

✅ Challenge: API Gateway returned an error when calling Lambda.

🔍 Diagnosis: API Gateway permissions were misconfigured.

🛠 Solution: Updated the IAM role to allow API Gateway to invoke Lambda.

✅ Challenge: Lambda function execution timed out.

🔍 Diagnosis: Default timeout setting was too low.

🛠 Solution: Increased Lambda timeout settings for stable execution.

## Results & Impact

- Successfully deployed an API endpoint that dynamically returns responses from Lambda.

- Gained hands-on experience troubleshooting AWS API integrations.

- Validated function execution through logs using CloudWatch.

## Next Steps

- Implement more API methods (GET, POST).

- Explore securing API Gateway with IAM authentication.

- Automate deployment with AWS CloudFormation or Terraform.