Serverless Lead Capture Web Application

Professional web presence. Zero maintenance.
Event Management Website .

By Monish Kumar V

Amazon S3

Amazon API Gateway

AWS Lambda

Amazon DynamoDB

Architectural Overview and Frontend Implementation

- **Project Goal** : To demonstrate a modern, scalable, and cost-effective method for hosting a static website and capturing user input using AWS Serverless technologies.
- **Key Focus Areas :**

1. AWS Service Integration.

2. Frontend Implementation (HTML/CSS/JS).

3. Securing Cross-Origin Requests (CORS).

Why Serverless?

Zero Maintenance

No servers to patch, monitor, or scale manually

Cost Efficient

Pay only for what you use, not idle capacity

Enterprise Security

AWS-managed encryption and compliance built-in

The Architecture

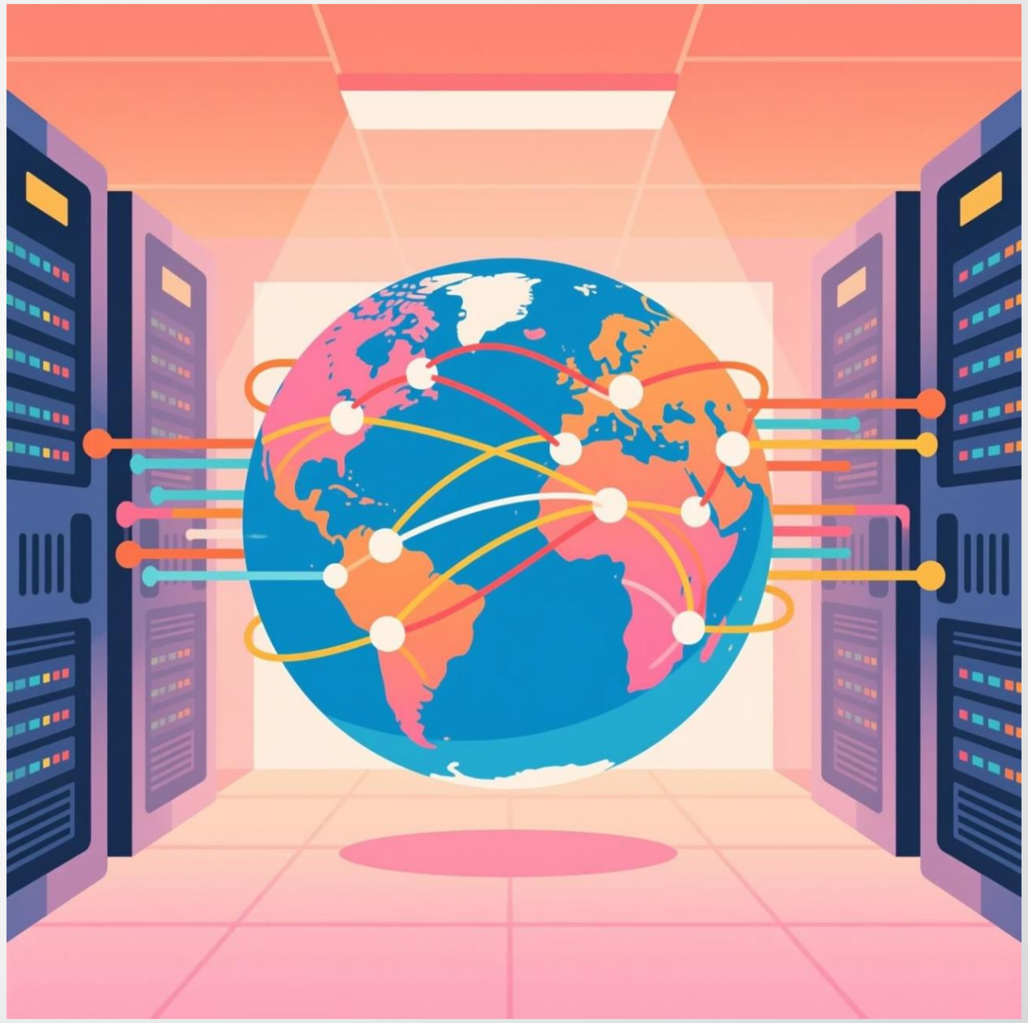
- **S3 Website**
- **API Gateway**
- **Lambda Function**
- **DynamoDB**

Static Site on Amazon S3

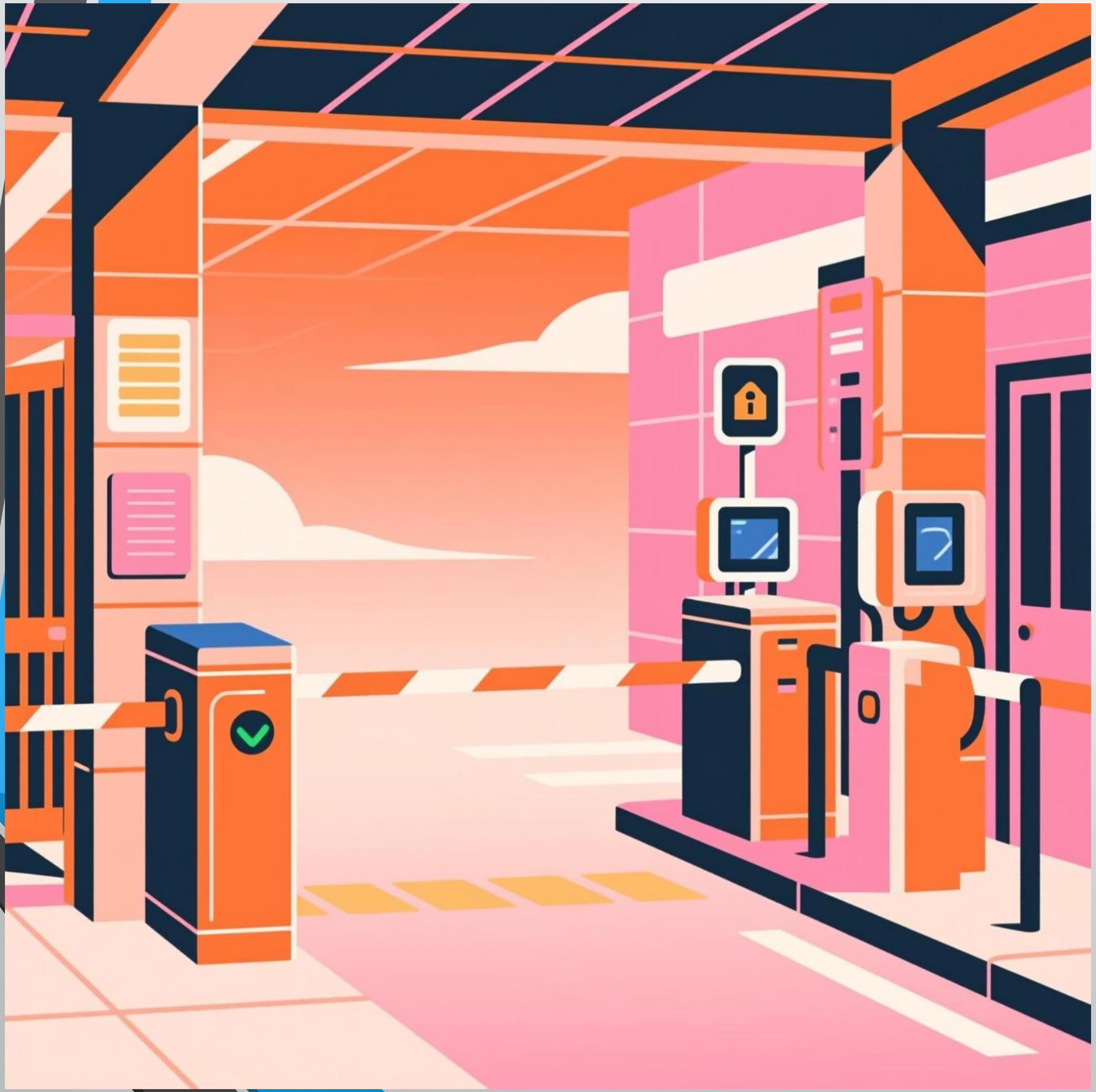
Fast & Reliable

Your website is hosted as a static site directly on S3. Lightning-fast content delivery globally with 99.99% uptime SLA.

- Automatic scaling for traffic spikes
- Global content distribution
- Built-in redundancy
- **Hosts all static assets (.html, .css, .js, images). Configured for public access as a website endpoint.**



API Gateway: Your Security Gatekeeper



Robust Protection

Provides a publicly accessible **REST API** endpoint for the contact form. Configures the essential **CORS headers** and serves as a firewall/router to Lambda.

- Cross-Origin Resource Sharing (CORS) policies

A security mechanism used by browsers to prevent a script loaded from one domain (S3 website) from interacting with a resource from a different domain (API Gateway).

- **Note : Essential because the frontend domain (S3) is different from the backend domain (API Gateway). Without correct CORS, the form request fails immediately.**

AWS Lambda : Backend Logic & Processing

Executes the Python code. Receives form data via Lambda Proxy Integration, validates the input, and prepares the record for the database.

On-Demand Execution

Python function runs only when a lead arrives—no idle costs

Data Validation

Automatically checks and sanitizes all incoming inquiries

Seamless Integration

Directly writes cleaned data to DynamoDB database

AWS IAM (Identity and Access Management) : Security & Permissions

Manages the **IAM Role** for the Lambda function, granting it necessary permissions (and only those permissions) to write data to the specific DynamoDB table.

AWSLambdaBasicExecutionRole :

Allows the Lambda function to create and write logs to the /aws/lambda/BlackTigerHandleContactFormFunction log group in CloudWatch.

AmazonDynamoDBFullAccess :

Grants permission to perform Read/Write operations (**dynamodb:PutItem, dynamodb:GetItem**, etc.) on all DynamoDB tables.

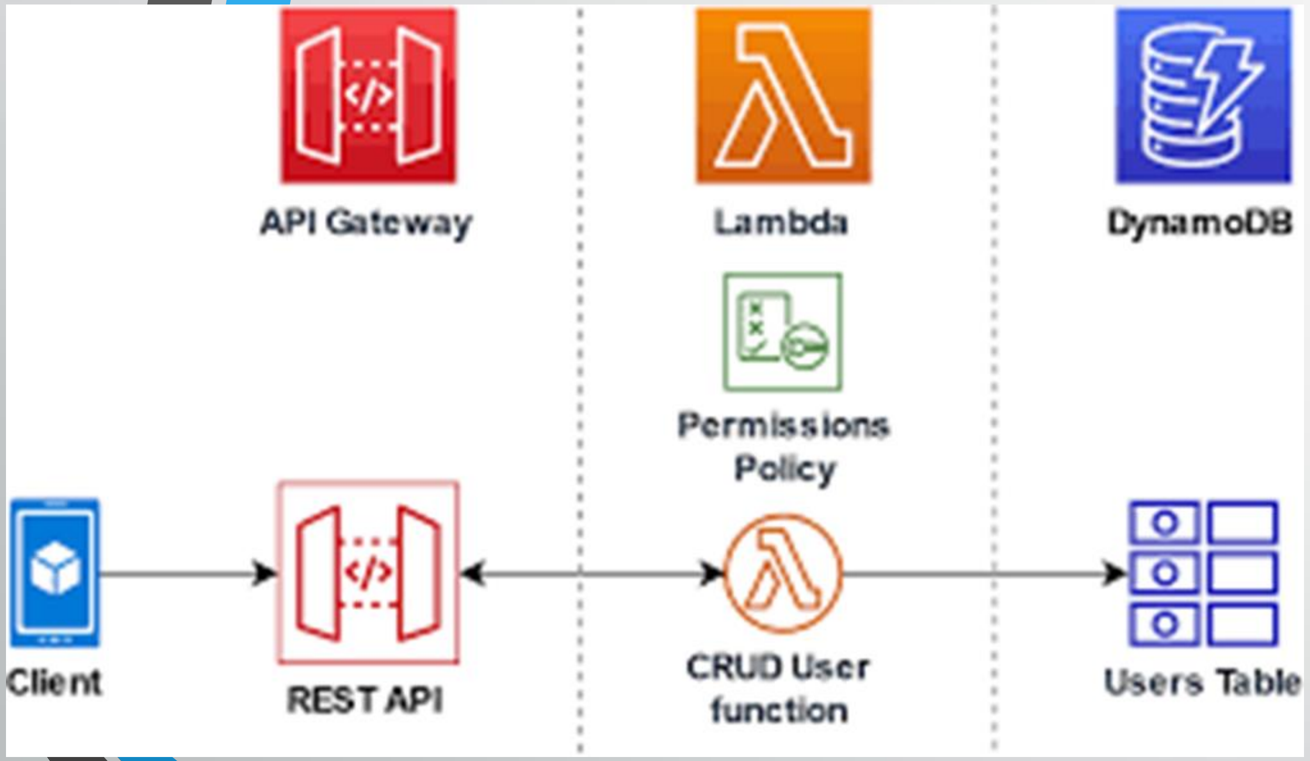
AWS DynamoDB: Data Storage

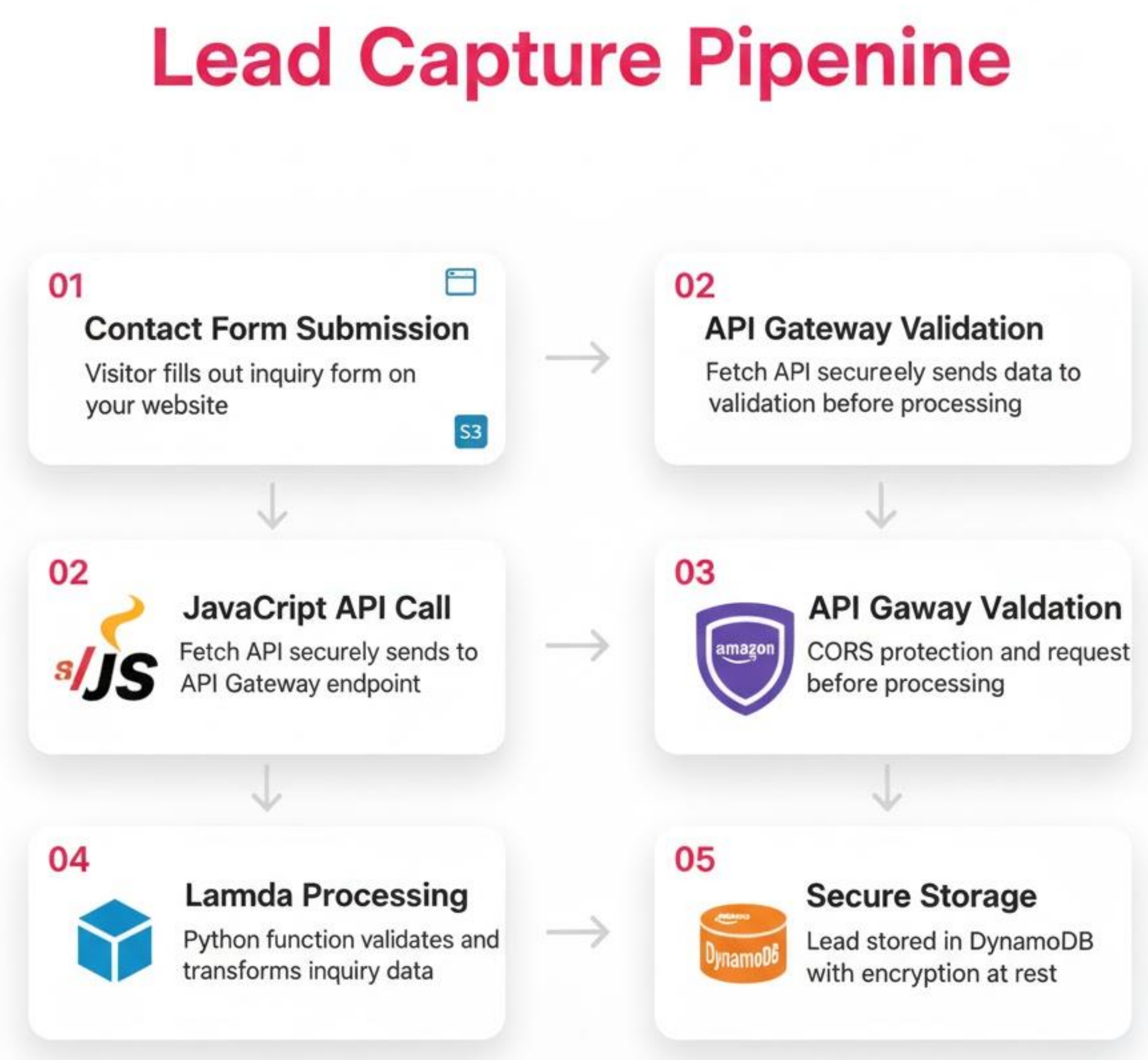
- **Scales infinitely with your growth. A fully managed NoSQL database used for storing captured client inquiries, ensuring high availability and scalability.**

Enterprise-Grade Storage

BlackTigerContactQueriesTable securely stores every lead with encryption at rest. Automatic backups ensure you never lose prospect data.

- Encrypted data storage
- Automatic daily backups
- Instant query retrieval





Lead Capture Pipenine

01 Contact Form Submission

Visitor fills out inquiry form on your website

S3

02 API Gateway Validation

Fetch API secureely sends data to validation before processing

02 JavaCript API Call

Fetch API securely sends to API Gateway endpoint

03 API Gaway Valdation

CORS protection and request before processing

04 Lamda Processing

Python function validates and transforms inquiry data

05 Secure Storage

Lead stored in DynamoDB with encryption at rest

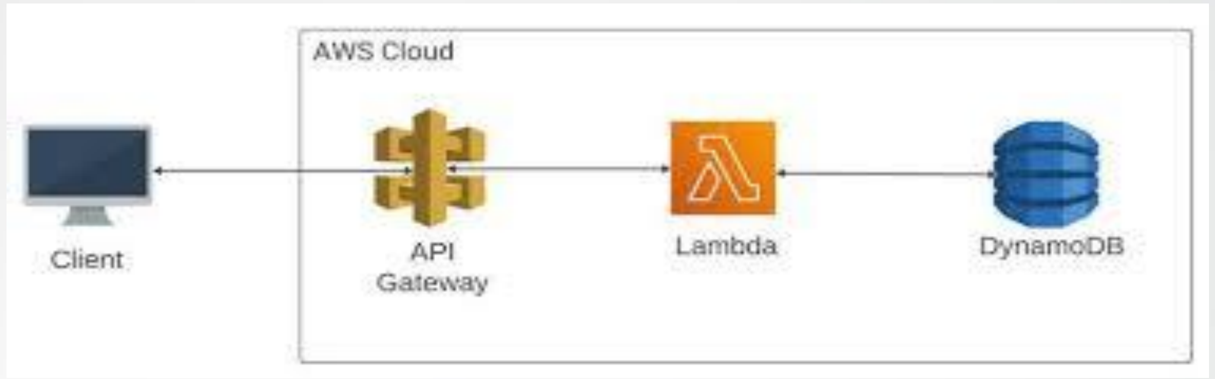
- **Frontend Languages :**

- **HTML5 :** Provides the basic **structure** and **content** of the web pages (index.html, contact.html)
- **CSS3 (Cascading Style Sheets):** Controls the **visual presentation** and **styling** of the website.
- **JavaScript :** Handles the **interactivity** and **API communication**.

- **Backend Language (Serverless Logic):**

- **Python (Executed by AWS Lambda):** Executes the serverless logic for **data processing** and **database interaction**.

Step to Configure



- **1. Hosting Setup (Amazon S3)**
- **Create S3 Bucket:** Create a new S3 bucket with a globally unique name (e.g., blacktigers-website-2025).
- **Enable Static Website Hosting:** Go to the **Properties** tab of the bucket, enable Static Website Hosting, set the **Index document** to index.html, and note the **Endpoint URL**.
- **Configure Bucket Policy:** Edit the **Permissions** tab and set a Bucket Policy to allow public read access (essential for a static website).
- **Upload Files:** Upload all frontend files (index.html, contact.html, style.css, script.js, and images) to the root of the bucket.

2. Database Creation (Amazon DynamoDB)

- **Create Table:** Create a new DynamoDB table named **BlackTigerContactQueriesTable**.

- **Define Primary Key:** Set the primary partition key to **queryId** (String). This will be used to uniquely identify each client inquiry.

3. Backend Logic (AWS Lambda)

- **Create Execution Role (IAM):** Create a new IAM role (e.g., BlackTigerLambdaDynamoRole) with a trust relationship for **lambda.amazonaws.com**. Attach the **AWSLambdaBasicExecutionRole** policy (for CloudWatch logs) and a **Custom Policy** granting **dynamodb:PutItem** permission restricted to the BlackTigerContactQueriesTable.

- **Create Lambda Function:** Create a new Lambda function using the **Python** runtime. Assign it the BlackTigerLambdaDynamoRole you just created.

- **Deploy Code:** Write and deploy the Python code that handles the API request, extracts the JSON data (name, email, query), generates a queryId, and uses the Boto3 library to perform the PutItem operation on the DynamoDB table.

4. API Endpoint (AWS API Gateway)

- **Create REST API:** Create a new **REST API**.
- **Create Resource:** Create a resource named /contact on the root of your API.
- **Create Method (POST):** Create a **POST** method under the /contact resource.
- **Integration Setup:** Set the Integration Type to **Lambda Function** and select your BlackTigerHandleContactFormFunction. Enable **Lambda Proxy Integration**.
- **Configure CORS:** Select the /contact resource and enable **CORS**. This is critical: set **Access-Control-Allow-Origin** to your S3 bucket's domain (or * if testing) to allow the frontend to communicate with the API.
- **Deploy API:** Deploy the API to a new stage (e.g., prod). Note the **Invoke URL**—you will paste this into your script.js.

5. Final Connection (Frontend Code Update)

- **Update JavaScript:** Open your **script.js** file.
- **Paste Invoke URL:** Replace the placeholder API_BASE_URL with the **Invoke URL** obtained from your API Gateway deployment stage.
- **Re-upload:** Upload the updated script.js file back to your S3 bucket.

Ready to Scale

Professional website. Automatic lead capture. Zero server maintenance. Enterprise security.

√ Live & Secure

Website hosted globally on S3

√ Lead Generation

Every inquiry automatically captured and stored

√ Peace of Mind

AWS manages infrastructure, you manage growth

Thank You