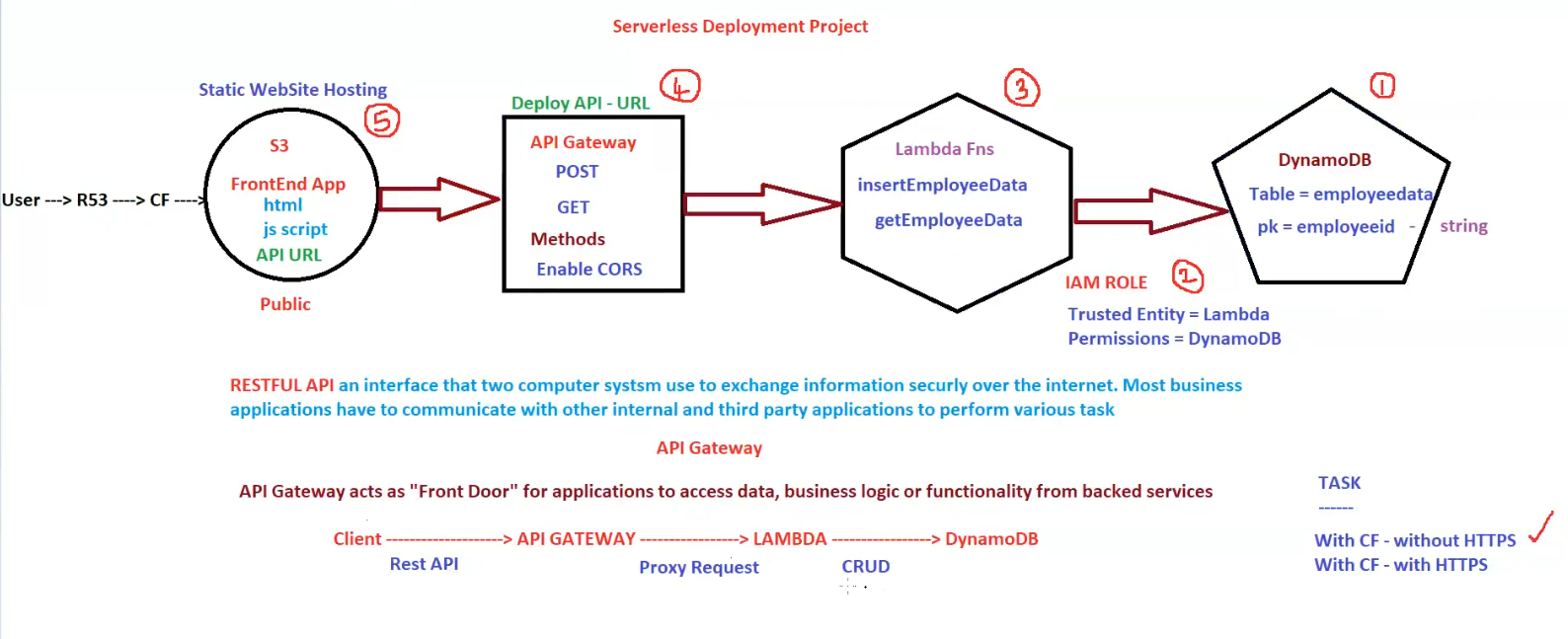
DevOps

[Serverless Deployment Project 2](#_Toc193690411)

# Serverless Deployment Project

****

**Topic: Serverless Deployment Project - Static Website Hosting with API Interaction**

**I. Overview**

* **Objective:** To deploy a static website with a serverless backend API to interact with a DynamoDB database.
* **Key Components:**
  + **Amazon S3 (Simple Storage Service):** For hosting the static website files (HTML, JavaScript).
  + **Amazon Route 53 (DNS):** For routing user requests to CloudFront.
  + **Amazon CloudFront (CDN):** For caching and delivering the website content globally.
  + **Amazon API Gateway:** For creating and managing the API endpoints.
  + **AWS Lambda:** For running the backend logic (CRUD operations).
  + **Amazon DynamoDB:** For storing the application's data.
  + **IAM Roles:** For managing permissions between services.

**II. Architecture and Workflow (Numbered Steps)**

1. **DynamoDB (1):**
   * **Table:** employeedata
   * **Primary Key (pk):** employeeid (string)
   * This is where the application's data is stored.
2. **IAM Role (2):**
   * **Trusted Entity:** Lambda function.
   * **Permissions:** Access to DynamoDB.
   * This role allows the Lambda function to interact with the DynamoDB table.
3. **Lambda Functions (3):**
   * insertEmployeeData**:** Handles POST requests to insert employee data into DynamoDB.
   * getEmployeeData**:** Handles GET requests to retrieve employee data from DynamoDB.
   * These functions contain the application's backend logic.
4. **API Gateway (4):**
   * **Deploy API - URL:** This is the endpoint URL that the frontend application uses to interact with the backend.
   * **Methods:**
     + **POST:** Used to insert employee data (calls insertEmployeeData Lambda function).
     + **GET:** Used to retrieve employee data (calls getEmployeeData Lambda function).
   * **Enable CORS:** Enables Cross-Origin Resource Sharing, allowing the frontend (served from a different origin) to make requests to the API.
   * API Gateway acts as the "front door" for the backend, routing requests to the appropriate Lambda functions.
5. **S3 (5):**
   * **Static Website Hosting:** S3 hosts the static website files (HTML, JavaScript).
   * **Frontend App:** The HTML and JavaScript code of the application.
   * **API URL:** The frontend code uses the API Gateway URL to make requests to the backend.
   * **Public:** The S3 bucket is configured for public access to serve the static website.

**III. User Interaction and Data Flow**

* **User -> Route 53 -> CloudFront:** The user accesses the website via Route 53, which routes the request to CloudFront for caching and delivery.
* **Frontend App (HTML, JavaScript):** The frontend application is loaded in the user's browser.
* **Client Rest API -> API Gateway -> Lambda -> DynamoDB:** The frontend application makes API requests (POST or GET) to the API Gateway URL. API Gateway routes these requests to the appropriate Lambda function. The Lambda function then interacts with DynamoDB to perform the requested operation (insert or retrieve data).

**IV. Key Concepts**

* **RESTful API:** An interface for two computer systems to exchange information securely over the internet.
* **API Gateway as "Front Door":** API Gateway acts as a single entry point for accessing backend services, simplifying the architecture and improving security.
* **Proxy Request:** API Gateway acts as a proxy, forwarding requests to the Lambda functions.
* **CRUD:** Create, Read, Update, Delete operations on the database.

**V. Tasks**

* **With CloudFront - without HTTPS:** Configure CloudFront to serve the static website without HTTPS.
* **With CloudFront - with HTTPS:** Configure CloudFront to serve the static website with HTTPS (recommended for security).

**VI. Key Benefits of this Architecture**

* **Serverless:** No need to manage servers, reducing operational overhead.
* **Scalability:** AWS services automatically scale to handle traffic spikes.
* **Cost-Effective:** Pay-as-you-go pricing model.
* **Performance:** CloudFront provides fast content delivery through caching.
* **Security:** IAM roles and API Gateway provide secure access to backend resources.