Serverless Resume Parser and Portfolio Website

This project demonstrates a serverless architecture for processing resumes and showcasing a developer portfolio. It leverages several AWS services for scalability, efficiency, and security.

Project Overview

This application consists of a frontend portfolio website and a backend resume parsing system. Users upload resumes via the website, which are then processed by a serverless Lambda function using Amazon Textract. Extracted data is stored in DynamoDB and visualized in a QuickSight dashboard. The entire infrastructure is managed using a CI/CD pipeline.

Architecture

The architecture is divided into three main components:

1. Frontend (Portfolio Website):

*   **Technology:** HTML, CSS, JavaScript (React recommended), S3, CloudFront.
*   **Functionality:** Displays Yanga's portfolio, skills, and a resume upload form.
*   **Deployment:** Hosted on S3, served via CloudFront. Automated deployment via CodePipeline and CodeBuild.

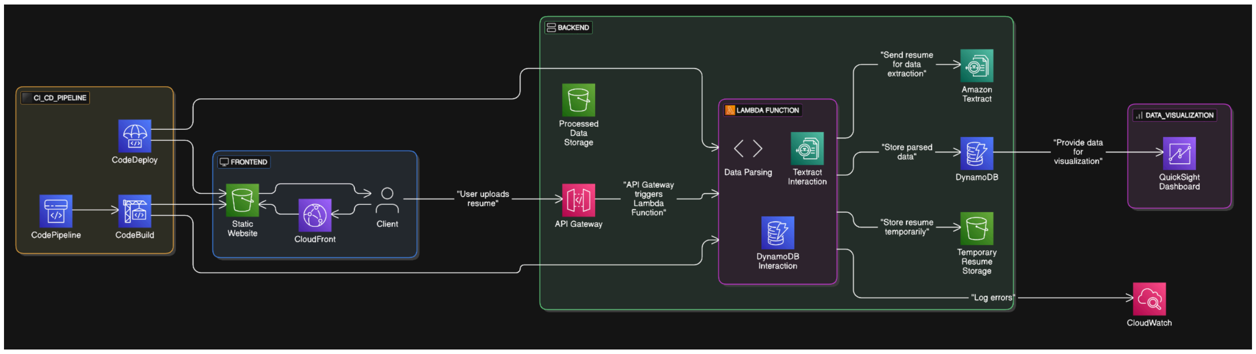
2. Backend (Resume Processing):

*   **Technology:** API Gateway, Lambda (Python), Amazon Textract, S3 (temporary storage), S3 (processed data storage), DynamoDB.
*   **Functionality:** Receives resume uploads, extracts data using Textract, processes the data, and stores it in DynamoDB.
*   **Deployment:** Lambda function deployment is automated via CodePipeline and CodeBuild. S3 buckets are managed via CloudFormation (recommended).

3. Data Visualization (QuickSight Dashboard):

*   **Technology:** Amazon QuickSight.
*   **Functionality:** Visualizes key metrics extracted from resumes (skill frequency, experience, etc.).
*   **Deployment:** Dashboard is manually created and linked to the DynamoDB table.

(Diagram)
[Insert a clear diagram of the architecture here. This should clearly show the flow of data and the interaction between different components.]



Functionality

1. **Resume Upload:** Users upload resumes via a form on the portfolio website. The form uses the Fetch API or similar.
2. **API Gateway Trigger:** The upload triggers an API Gateway endpoint.
3. **Lambda Function Execution:** The API Gateway invokes a Lambda function.
4. **Resume Processing:** The Lambda function:
    *   Temporarily stores the resume in an S3 bucket.
    *   Uses Amazon Textract to extract text and data.
    *   Parses the extracted data to identify skills, experience, education, etc.
    *   Stores the structured data in DynamoDB.
    *   Deletes the temporary resume file from S3.
5. **Data Visualization:** QuickSight dashboard displays insights from the DynamoDB data.

Deployment Steps

1. **Clone the Repository:** Clone this repository to your local machine: `git clone <repository_url>`

2. **Install Dependencies:** Install the necessary Node.js and Python packages (refer to `package.json` and `requirements.txt`).

3. **Build Frontend:** Build the React application (if applicable): `npm run build`

4. **Deploy Frontend:** Deploy the built frontend files to the S3 bucket for website hosting. Use the AWS CLI or console.

5. **Deploy Backend:** Deploy the Lambda function to AWS. Ensure the correct IAM role is attached and environment variables are set.

6. **Create DynamoDB Table:** Create the `ParsedResumes` table with the specified schema.

7. **Create API Gateway Endpoint:** Set up the API Gateway endpoint and integrate it with your Lambda function.

8. **Configure S3 Event Notification:** Set up an S3 event notification to trigger the Lambda function when new resumes are uploaded.

9. **Create QuickSight Dashboard:** Create the QuickSight dashboard and connect it to your DynamoDB table.

10. **CI/CD Pipeline (Optional):** Configure a CodePipeline pipeline to automate build and deployment (refer to the `buildspec.yml` file for build instructions).

Technology Stack

*   **Frontend:** React (or similar), HTML, CSS, JavaScript, S3, CloudFront
*   **Backend:** API Gateway, Lambda (Python), Amazon Textract, S3, DynamoDB
*   **Data Visualization:** Amazon QuickSight
*   **CI/CD:** AWS CodePipeline, CodeBuild

Contributing

Contributions are welcome! Please open an issue or submit a pull request.

License

[Specify your license here (e.g., MIT License)]