AWS underneath and Integration with React

This blog will go through how AWS is set up and what services were used to both deploy the website on domain name and to accept contact messages from viewers.

The AWS wiring that is under this project is for the functionality that will allow views of my website to contact me.

This includes storing their message and their contact information in DynamoDB and sending the same data to me via email. The AWS services for this functionality include ***API Gateway***, ***Lambda***, ***AWS SNS***, ***AWS SQS***, ***DynamoDB***.

The API gateway has an HTTP post request method defined. This gateway has *proxy integration* that will immediately relay the post request to a ***Lambda function***.

```
import { SNSClient, PublishCommand } from "@aws-sdk/client-sns";
import { SQSClient, SendMessageCommand } from "@aws-sdk/client-sqs";

const sqs_client = new SQSClient({ apiVersion: "2012-11-05" });
const sns_client = new SNSClient({ apiVersion: "2010-03-31" });

export const handler = async (event) => {
    const body = JSON.parse(event.body)
    console.log(body, "my event")
    const sqs_command = new SendMessageCommand({
        MessageBody: JSON.stringify(body),
        QueueUrl: "<sqs-link>"
    });
    await sqs_client.send(sqs_command);

    const sns_command = new PublishCommand({
        Message: `Contact from Kalebtsegaye.com
                Name: ${body.name}
                E-mail: ${body.email}
                Title: ${body.title}
                Phone: ${body.phone ? body.phone : 'N/A'}
                Message: ${body.message}
        `,
        TopicArn: "<sns-topic-arn>"
    });
    await sns_client.send(sns_command);

    const response = {
        statusCode: 200,
        headers: {
            'Content-Type': 'application/json',
            'Access-Control-Allow-Origin': '*'
        },
        body: JSON.stringify(`Contact Successfull`),
    };
    return response;
};
```

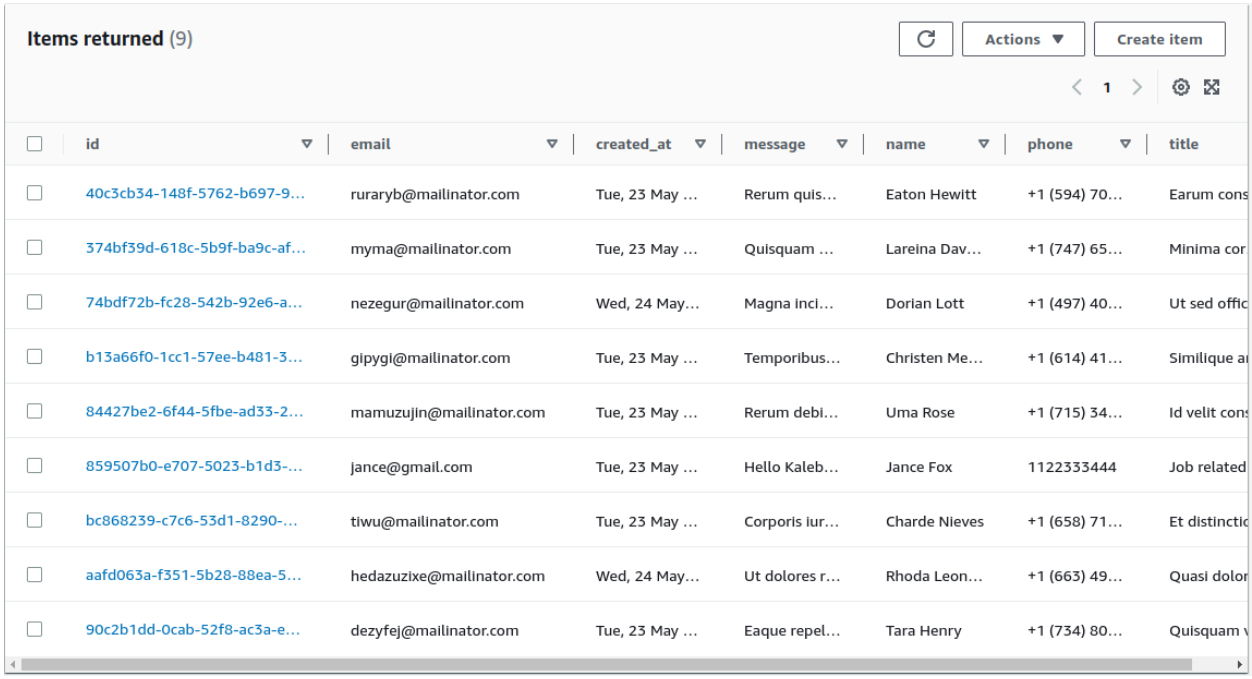
This function pushes a message to a ***SQS*** queue and sends me an email with the content of the form through an ***SNS*** topic.

The second Lambda will be triggered by an ***SQS*** event and will parse the message data to save them to ***DynamoDB***.

```
import {
    DynamoDBClient,
    PutItemCommand,
} from "@aws-sdk/client-dynamodb";

const dynamodb = new DynamoDBClient({
    apiVersion: "2012-08-10"
});

export const handler = async(event, context) => {
    const body = JSON.parse(event.Records[0].body)
    const saveParameters = {
        TableName: 'kalebtsegaye-com',
        Item: {
            "id": {},
            "name": {},
            "email": {},
            "title": {},
            "phone": {},
            "message": {},
            "created_at": {}
        }
    };
    const command = new PutItemCommand(saveParameters);
    await dynamodb.send(command);
    const response = {
        statusCode: 200,
        body: 'success'
    };

    return response;
};
```

The project structure is similar to the given requirement.

I have attached the list of items in ***DynamoDB*** and email I received during the test of the project.



Items returned (9)

Actions ▼ | Create item

1

id	email	created_at	message	name	phone	title
40c3cb34-148f-5762-b697-9…	rurarybb@mailinator.com	Tue, 23 May …	Rerum quis…	Eaton Hewitt	+1 (594) 70…	Earum cons
374bf39d-618c-5b9f-ba9c-af…	myma@mailinator.com	Tue, 23 May …	Quisquam …	Lareina Dav…	+1 (747) 65…	Minima cor
74bdf72b-fc28-542b-92e6-a…	nezegur@mailinator.com	Wed, 24 May…	Magna inci…	Dorian Lott	+1 (497) 40…	Ut sed offic
b13a66f0-1cc1-57ee-b481-3…	gipygi@mailinator.com	Tue, 23 May …	Temporibus…	Christen Me…	+1 (614) 41…	Similique a
84427be2-6f44-5fbe-ad33-2…	mamuzujin@mailinator.com	Tue, 23 May …	Rerum debi…	Uma Rose	+1 (715) 34…	Id velit cons
859507b0-e707-5023-b1d3-…	jance@gmail.com	Tue, 23 May …	Hello Kaleb…	Jance Fox	1122333444	Job related
bc868239-c7c6-53d1-8290-…	tiwu@mailinator.com	Tue, 23 May …	Corporis iur…	Charde Nieves	+1 (658) 71…	Et distinctio
aafd063a-f351-5b28-88ea-5…	hedazuzixe@mailinator.com	Wed, 24 May…	Ut dolores r…	Rhoda Leon…	+1 (663) 49…	Quasi dolor
90c2b1dd-0cab-52f8-ac3a-e…	dezyfej@mailinator.com	Tue, 23 May …	Eaque repel…	Tara Henry	+1 (734) 80…	Quisquam v

Notify Kaleb Tsegaye <no-reply@sns.amazonaws.com>
to me

Contact from Kalebtsegaye.com

Name: Charde Nieves
E-mail: tiwu@mailinator.com
Title: Et distinctio Et om
Phone: +1 (658) 716-2564
Message: Corporis iure duis q

•••

Notify Kaleb Tsegaye <no-reply@sns.amazonaws.com>
to me

Contact from Kalebtsegaye.com

Name: Rhoda Leonard
E-mail: hedazuzixe@mailinator.com
Title: Quasi doloribus non
Phone: +1 (663) 494-3746
Message: Ut dolores rerum nis

•••

Notify Kaleb Tsegaye <no-reply@sns.amazonaws.com>
to me

Contact from Kalebtsegaye.com

Name: Dorian Lott
E-mail: nezegur@mailinator.com
Title: Ut sed officiis cons
Phone: +1 (497) 403-1748
Message: Magna incidunt sit

•••

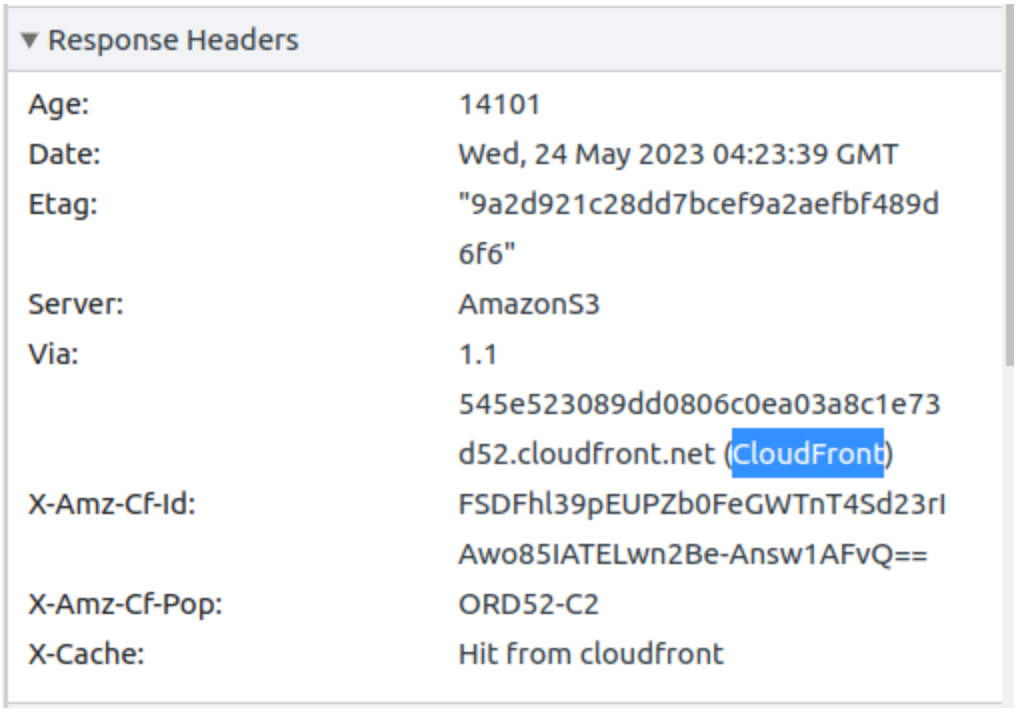
Route53 and CloudFront

While the React side was under development, I purchased a domain name from a service called ***IONOS***. The domain I purchased is https://kalebtsegaye.com
An ***S3*** bucket contains the production build files of react to serve them through ***Static website hosting*.**

I initially created a hosted zone on ***Route53*** and pasted the NS record values to the IONOS console of my domain name. That enabled me to control the deployment to my domain from AWS. So, at this point, going to https://kalebtsegaye.com serves the react application directly from my ***S3*** bucket.

The next step was requesting a certificate from Amazon Certificate Manager and creating ***Route53*** CNAME records.
Next I created ***CloudFront*** distributions that will be invalidated everytime a new version is pushed to the remote repository.

Here is the response header in https://kalebtsegaye.com on the web contents.
It shows that the site's Server is ***AmazonS3*** but it is being served from ***CloudFront***.


Response Headers
Age:
14101
Date:
Wed, 24 May 2023 04:23:39 GMT
Etag:
"9a2d921c28dd7bcef9a2aefbf489d6f6"
Server:
AmazonS3
Via:
1.1 545e523089dd0806c0ea03a8c1e73d52.cloudfront.net (CloudFront)
X-Amz-Cf-Id:
FSDFhl39pEUPZb0FeGWTnT4Sd23rIAwo85IATELwn2Be-Answ1AFvQ==
X-Amz-Cf-Pop:
ORD52-C2
X-Cache:
Hit from cloudfront