AWS Deployment Report

Syed Faiq Haider Naqvi
2022562

Overview

This report details the deployment of a static web application using AWS services such as **AWS Amplify, Lambda, API Gateway, and DynamoDB**. The application tracks website visits and displays them on the webpage. Additionally, a serverless function sends a message when the website loads.

I. Changes Made to the GitHub Repository

1. index.html (Updated for visit count display)

Added an HTML element to display the visit count:

```
<h2 id="visitCount">Loading...</h2>
```

2. script.js (Updated to fetch visit count and trigger alert)

Modified the script to fetch visit count from the API Gateway:

```
const API_URL = "https://pbc5vzoite.execute-api.eu-north-1.amazc

async function fetchVisitCount() {
    try {
        const response = await fetch(API_URL);
        const data = await response.json();

        alert(data.message); // Displays "Hello from the server!

        document.getElementById("visitCount").innerText = "Visit
    } catch (error) {
        console.error("Error fetching visit count:", error);
    }
}

fetchVisitCount();
```

3. GitHub Commit & Push

After making changes, the following commands were used:

git add index.html script.js

git commit -m "Updated API integration for visit count and alert message"

git push origin main

AWS Amplify automatically deployed the updated web application.

II. Important Links

- **Live Web App:** https://main.d1ehg6mwjoboan.amplifyapp.com/
- **API Gateway Invoke URL:** https://pbc5vzoite.execute-api.eu-north-1.amazonaws.com/prod/visitCount

III. Step-by-Step Breakdown

1. Create Web App (5 minutes)

- Used **AWS Amplify Console** to deploy static web resources.
- Connected the GitHub repository.
- Enabled automatic deployments when code is updated.
- **Outcome:** Website is hosted and accessible.

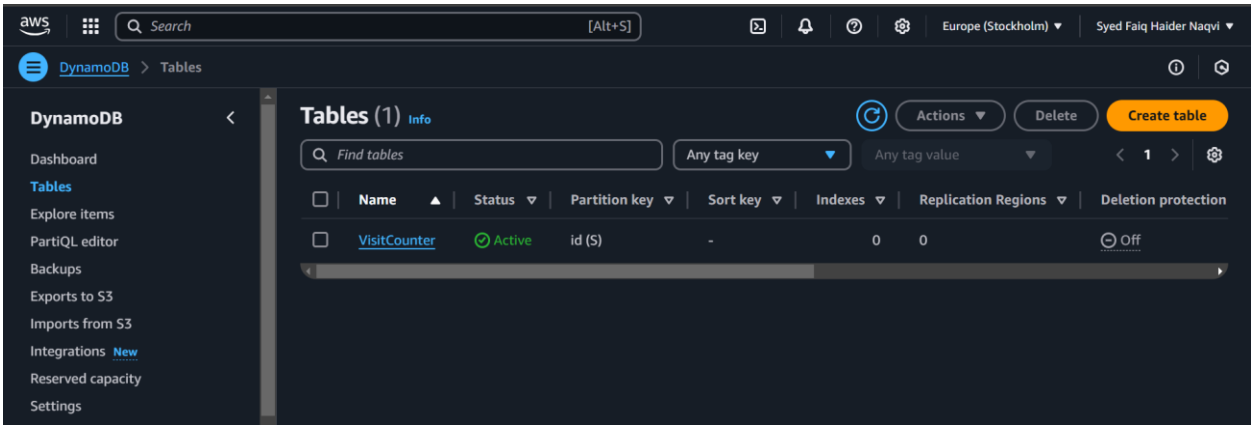
2. Build Serverless Function (5 minutes)

- Created an **AWS Lambda** function (updateVisitCount).
- Function increments the visit count stored in **DynamoDB** and returns a message.
- **Lambda Code:**

```python
import json
import boto3

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('VisitCounter')

def lambda_handler(event, context):
    response = table.get_item(Key={'id': 'visitCount'})
    visit_count = response['Item']['visitCount']
    visit_count += 1

    table.update_item(
        Key={'id': 'visitCount'},
        UpdateExpression='SET visitCount = :val',
        ExpressionAttributeValues={':val': visit_count}
    )

    return {
        'statusCode': 200,
        'body': json.dumps({
            'visitCount': visit_count,
            'message': 'Hello from the serverless function'
        })
    }
```

- **Outcome:** Function created and tested in AWS Lambda.

3. Create Data Table (10 minutes)



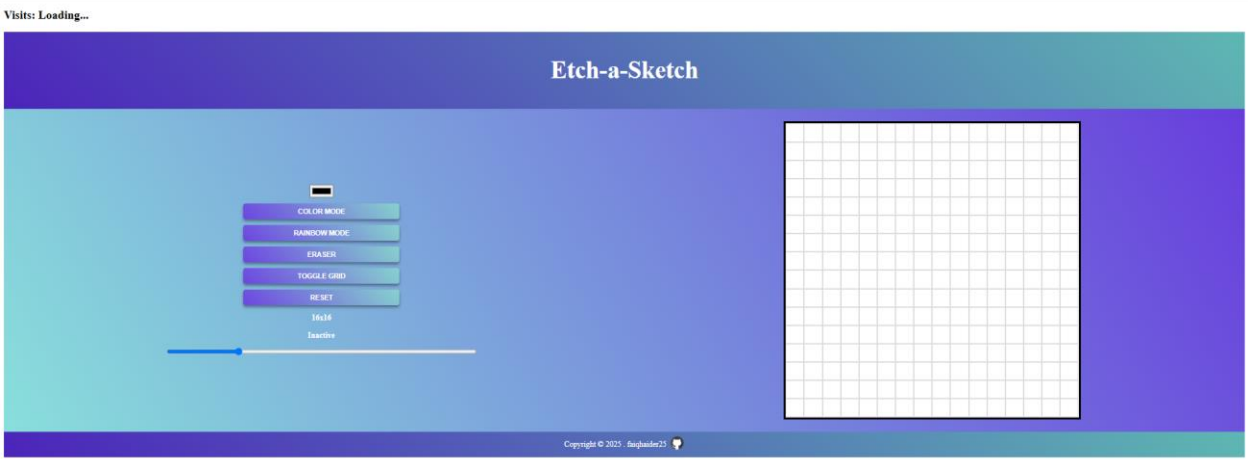
- **DynamoDB Table:** VisitCounter
- Attributes:
    - **id (String, Partition Key)** → Set to visitCount
    - **visitCount (Number)** → Initially set to 0

- **Outcome:** Table created and populated.

4. Link Serverless Function to Web App (5 minutes)

- Created an **API Gateway**.
- Added a **GET /visitCount** route.
- Integrated the API Gateway route with the Lambda function.
- Deployed the API.
- **Outcome:** API Gateway exposes the Lambda function.

5. Add Interactivity to Web App (5 minutes)



- Updated script.js to call API Gateway and update visit count.
- **Outcome:** The webpage dynamically displays the updated visit count and shows an alert.

6. Clean Up Resources (To Be Done After Demo)

- This step will be completed after the TA demo.
- Includes deleting the **Lambda function, API Gateway, DynamoDB table, and Amplify deployment**.

IV. Current Issues & Next Steps

- **Issue:** API Gateway currently returns {"message":"Not Found"}.
- **Next Steps:**
    - Verify API Gateway routes.

- Ensure proper connection between API Gateway and Lambda.
- Debug API Gateway and Lambda permissions.

Conclusion

This report outlines the full deployment process of a serverless web application using AWS Amplify, Lambda, API Gateway, and DynamoDB. The final step (cleaning up resources) will be completed after the TA demo.

Status: Web app deployed, API connection needs debugging.