Notes

AWS Project Documentation

URL Shortener using AWS Services

@sAnandha

Notes

URL SHORTENER USING AWS SERVICES

1. **Introduction**

A URL shortener is a tool that converts long URLs into shorter, more manageable links. This project leverages AWS serverless architecture to create a scalable and cost-effective URL shortener. Users enter a long URL, and the system generates a short URL that redirects to the original address.

2. **Services Used**

We use AWS services to ensure a reliable, scalable, and cost-efficient solution:

✅ AWS Lambda - Handles backend logic for generating and retrieving short URLs

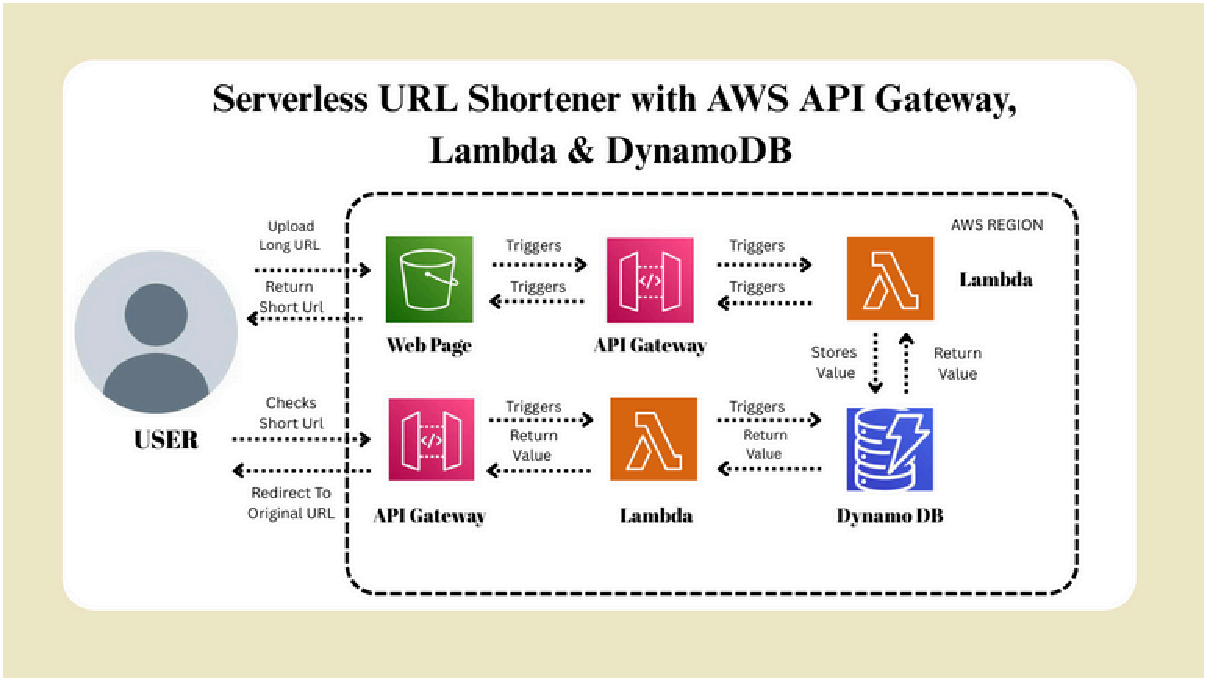
✅ API Gateway - Acts as a bridge between the frontend and backend

✅ DynamoDB - Stores the original and shortened URLs

✅ S3 - Hosts the frontend HTML, CSS, and JavaScript

Notes

ARCHITECTURE DIAGRAM :



Architecture Overview :

The system follows a serverless design using AWS:

1️⃣ **User Input**: A user enters a long URL in the frontend (HTML & JavaScript).

2️⃣ **API Gateway**: Sends the request to AWS Lambda.

3️⃣ **Lambda** (Shortening):

- Generates a unique short code.
- Stores the mapping (short code → original URL) in DynamoDB.

4️⃣ **Lambda** (Redirection):

- When a user accesses a short URL, Lambda fetches the original URL from DynamoDB.
- Redirects the user to the original site.

5️⃣ **Frontend** (S3/CloudFront): Hosts the user interface for entering URLs and receiving short links.

Notes

Steps to deploy the URL shortener application using AWS services :

1. **Backend Setup**

1.1 **Create a DynamoDB Table**

- Open AWS DynamoDB.
- Create a table named URLShortener.
- Set the Primary Key as shortCode (String).
- This table stores the short codes and their corresponding original URLs.

1.2 **Create AWS Lambda Functions**

- Go to AWS Lambda and create two functions:
  a. **Shorten URL Function**: Generates a short code and saves the original URL in DynamoDB.
  b. **Redirect URL Function**: Retrieves the original URL from DynamoDB and redirects the user.
- Set the runtime as Python or Node.js.

1.3 **Attach IAM Permissions to Lambda**

- Create an IAM Role with DynamoDB Full Access and attach it to both Lambda functions.
- The role should allow the functions to read and write to the DynamoDB table.

Notes

1.4 **Create an API Gateway**

- Open Amazon API Gateway.
- Create a new REST API.
- Add the following endpoints:
  - POST /shorten → Connect to the Shorten URL Lambda Function.
  - GET /{short_code} → Connect to the Redirect URL Lambda Function.
- Enable CORS (Cross-Origin Resource Sharing) for both endpoints.
- Deploy the API and note the API endpoint URL.

2. **Frontend Setup**

2.1 **Create an HTML Page**

- Design a simple form where users can enter a long URL.
- Include a button to generate a short URL.
- Display the generated short URL to the user.

2.2 **Host the Frontend on Amazon S3**

- Create an S3 bucket with public access enabled.
- Enable static website hosting.
- Upload the HTML file to the bucket.
- Secure the S3 BucketSet an S3 bucket policy to allow public read access.

Notes

3. Testing the URL Shortener

3.1 Test Shortening a URL

- Open the hosted webpage in a browser.
- Enter a long URL and click "Shorten".
- The API should return a shortened URL.

3.2 Test Redirecting a Shortened URL

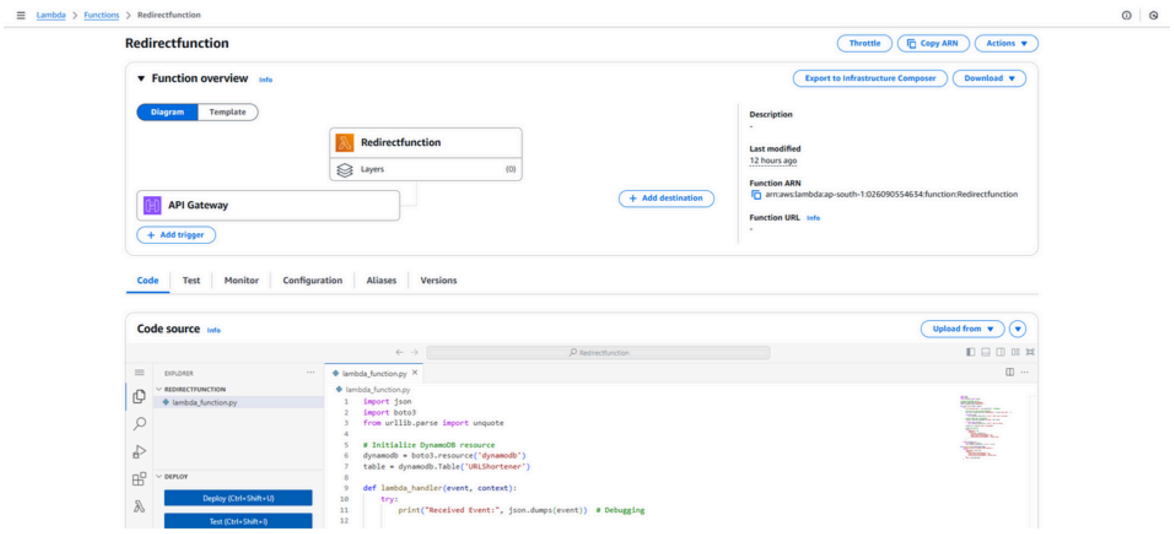
- Click on the shortened URL.
- It should redirect to the original long URL.

Conclusion

- The URL shortener is now live and functional.
- Users can shorten URLs and access them via unique short links.
- AWS provides scalability, security, and cost efficiency.

Notes

SAMPLE OUTPUT :



Redirectfunction

Description
-
Last modified
12 hours ago
Function ARN
arn:aws:lambda:ap-south-1:026090554634:function:Redirectfunction
Function URL
-

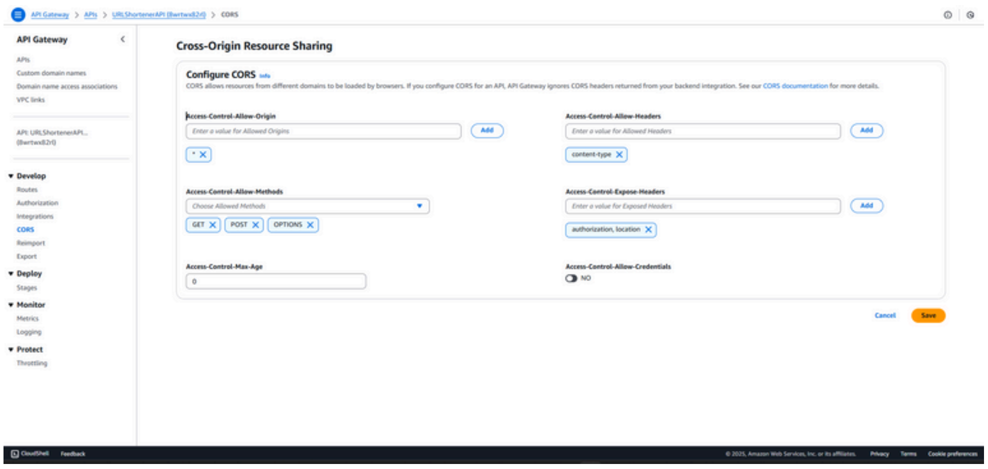
lambda_function.py

```python
import json
import boto3
from urllib.parse import unquote

# Initialize DynamoDB resource
dynamodb = boto3.resource("dynamodb")
table = dynamodb.Table('URLShortener')

def lambda_handler(event, context):
    try:
        print("Received Event:", json.dumps(event))  # Debugging

```



Cross-Origin Resource Sharing

Configure CORS

CORS allows resources from different domains to be loaded by browsers. If you configure CORS for an API, API Gateway ignores CORS headers returned from your backend integration. See our CORS documentation for more details.

Access-Control-Allow-Origin: *

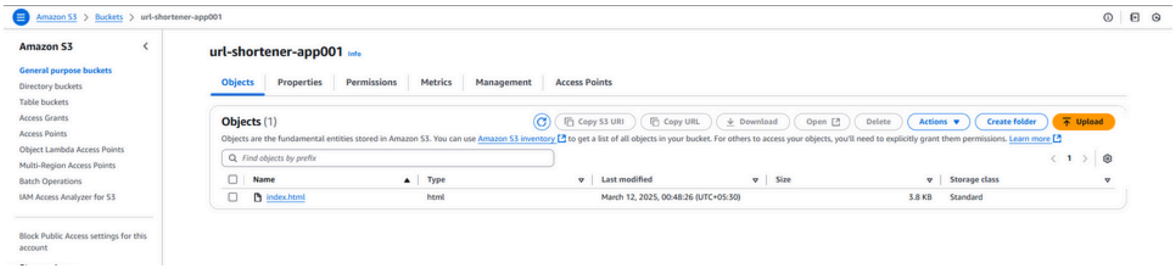
Access-Control-Allow-Headers: content-type

Access-Control-Allow-Methods: GET, POST, OPTIONS

Access-Control-Expose-Headers: authorization, location

Access-Control-Max-Age: 0

Access-Control-Allow-Credentials: NO

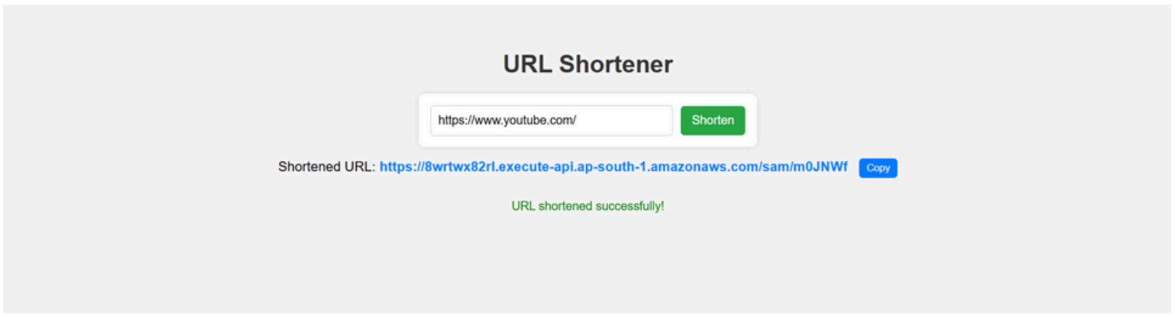


url-shortener-app001

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
index.html	html	March 12, 2025, 00:48:26 (UTC+05:30)	3.8 KB	Standard



URL Shortener

https://www.youtube.com/

Shortened URL: https://8wrtwx82rl.execute-api.ap-south-1.amazonaws.com/sam/m0JNWf

URL shortened successfully!

@sAnandha