Deployment Guide (Manual via Console)
URL Shortener

Overview

When a user enters a long URL, the system generates a short, unique identifier (like abc123). The user can then share this short URL. When someone clicks on it, the system looks up the identifier, finds the original URL, and redirects the visitor to that URL.
Example:

Original Long URL		Short URL
https://www.amazon.com/product/B08KH53NKR/ref	=	https://short.ly/abc123

Goals & Components Explained

1. Serverless URL Shortener

- **Serverless** means you don't manage servers — AWS handles it.
- The logic runs in **AWS Lambda**, which only executes when triggered (you only pay for usage).
- No EC2, no backend servers — making it **scalable and cost-effective**.

2. Frontend: Static Website

- The user interface (form to enter URL, display short URL) is hosted as a **static website** in **Amazon S3**.
- It might be a simple HTML/JavaScript app.
- You can enable **S3 static website hosting** and optionally use **CloudFront** to serve it globally.

3. Backend: URL Shortening & Redirection via Lambda

- When a user submits a long URL:
  - The frontend sends it to an **API Gateway endpoint**.
  - The request triggers an **AWS Lambda function**.
  - The Lambda function:
    - Generates a **short unique ID**.
    - Stores the mapping (short_id → long_url) in **DynamoDB**.
    - Returns the short URL to the frontend.
- When someone visits the short URL:
  - The short ID is passed to another **API Gateway + Lambda** endpoint.
  - Lambda looks up the short ID in DynamoDB.
  - If found, it **redirects** the user to the original long URL using a 301 redirect.

4. Data Storage: DynamoDB

- A **NoSQL database** to store mappings:
  - Partition key: short_id

    - Attribute: long_url
- Fast lookups, fully managed, and scales automatically.

Architecture Summary

[User] → [S3 Static Website] → [API Gateway] → [Lambda Function] → [DynamoDB]
↑
Short ID
↓
Redirect to long URL

AWS Services Used

Service	Purpose
Amazon S3	Host static frontend
API Gateway	Expose REST endpoints for frontend to call
AWS Lambda	Handle logic for URL shortening and redirection
Amazon DynamoDB	Store short URL to long URL mappings

Testing Goals

- Focus is on **building and testing the core logic**:
    - Submit long URL → receive short URL
    - Visit short URL → get redirected to long one

Architecture Overview (with Diagram)

Here's how the system works, visually:

System Flow

[User on Web Browser]
|
v
[Static Web Page on S3]
|
(Submit Long URL using JavaScript)
|
v
[API Gateway (POST)]
|
v
[Lambda Function #1: Shorten URL]
|
v
[DynamoDB: Save short_id + long_url]

|
v
← Return short URL to frontend

Now when the user visits the short URL:

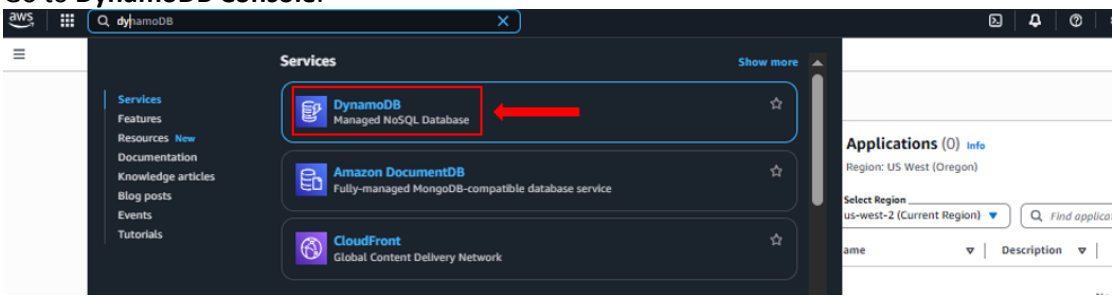
[User clicks short URL: /abc123]
|
v
[API Gateway (GET) with path /abc123]
|
v
[Lambda Function #2: Redirect]
|
v
[DynamoDB: Lookup abc123 → long_url]
|
v
← HTTP 301 Redirect to original URL

📘 URL Shortener Project Tutorial with Screenshots (AWS Serverless)

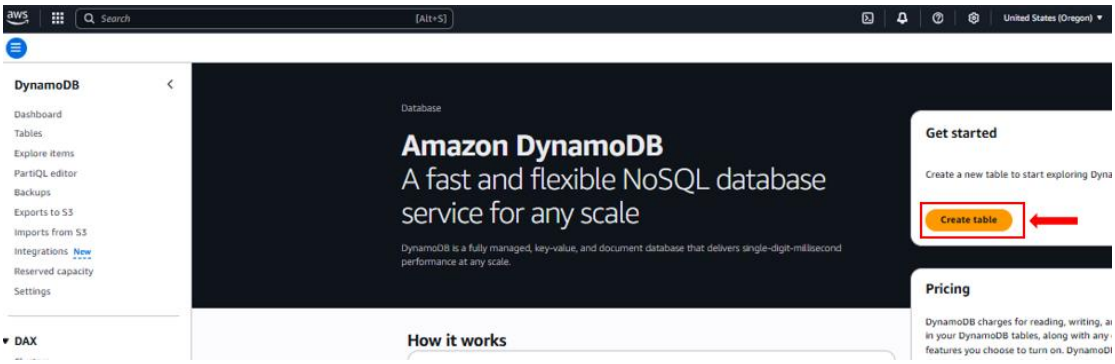
This guide walks you through creating a **serverless URL shortener** using AWS services: **Lambda, DynamoDB**, and **S3** — with visuals for every step.

✅ Step 1: Create a DynamoDB Table

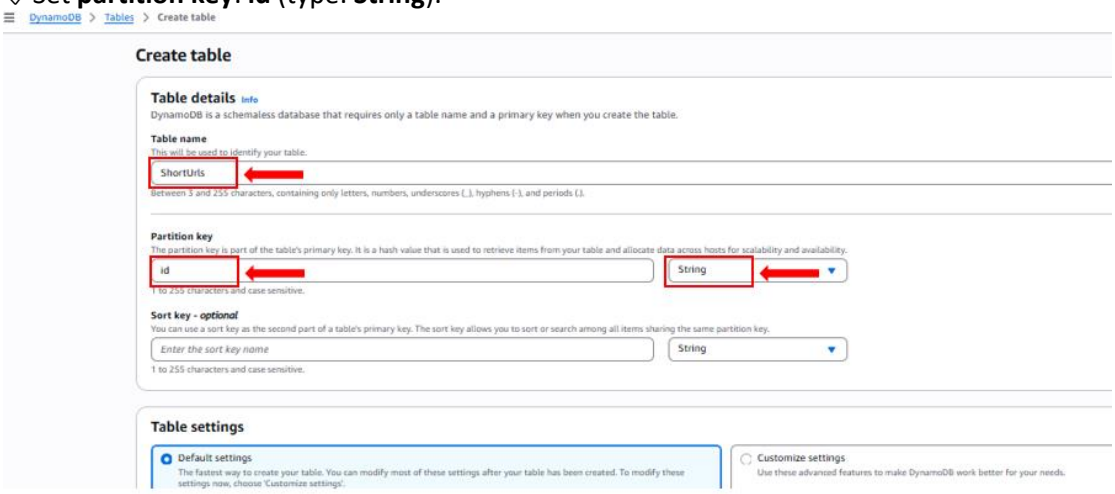
1. Go to **DynamoDB Console**.



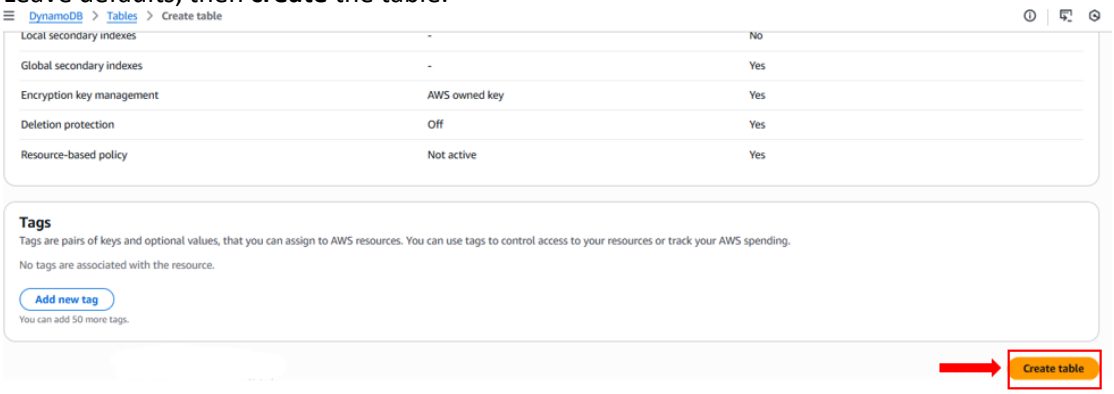
2. Click **Create table** .



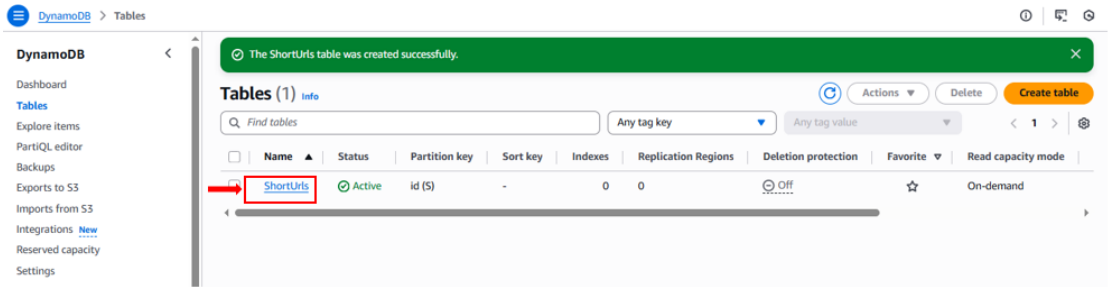
3. Set these values :
   ✅Table Name "**ShortUrls**".
   ✅Set **partition key**: **id** (type: **String**).



4. Leave defaults, then **create** the table.

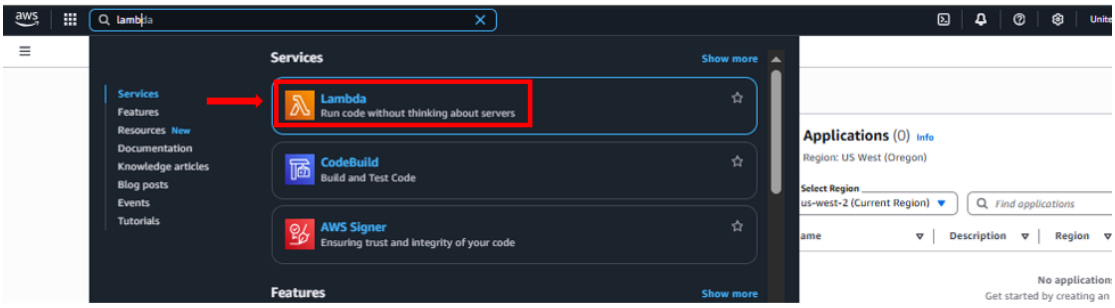


5. Successfully created "**ShortUrls**" table.

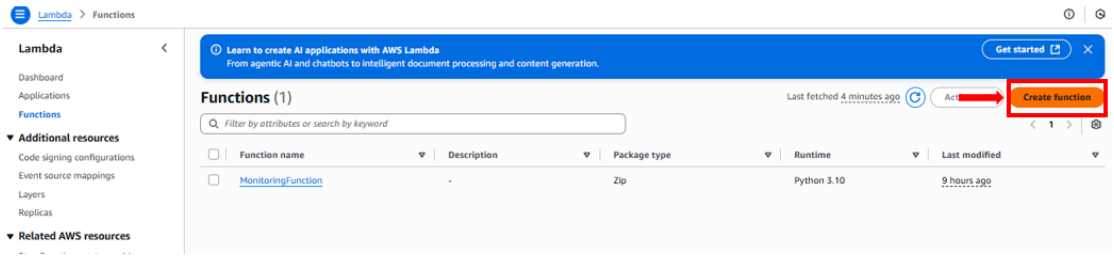


✅ Step 2: Create the Lambda Function

1. Go to **Lambda Console.**

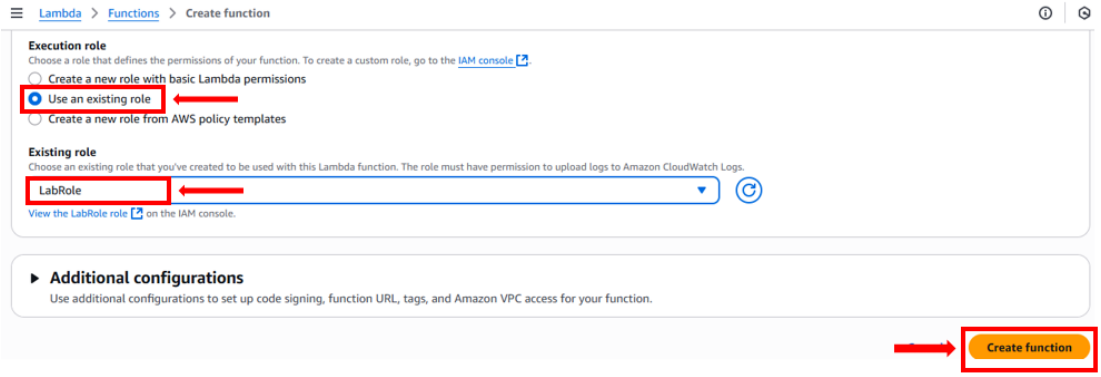


2. Click **Create Function** .

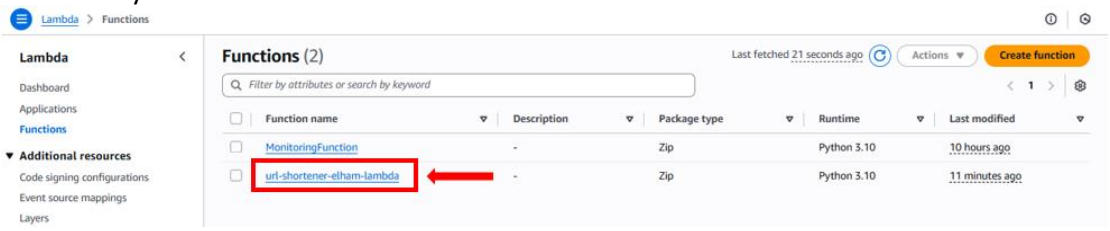


3. Choose:
    1. ✅ **Author from scratch**.
    2. Name: **url-shortener-elham-lambda**.
    3. Runtime: **Python 3.10**.

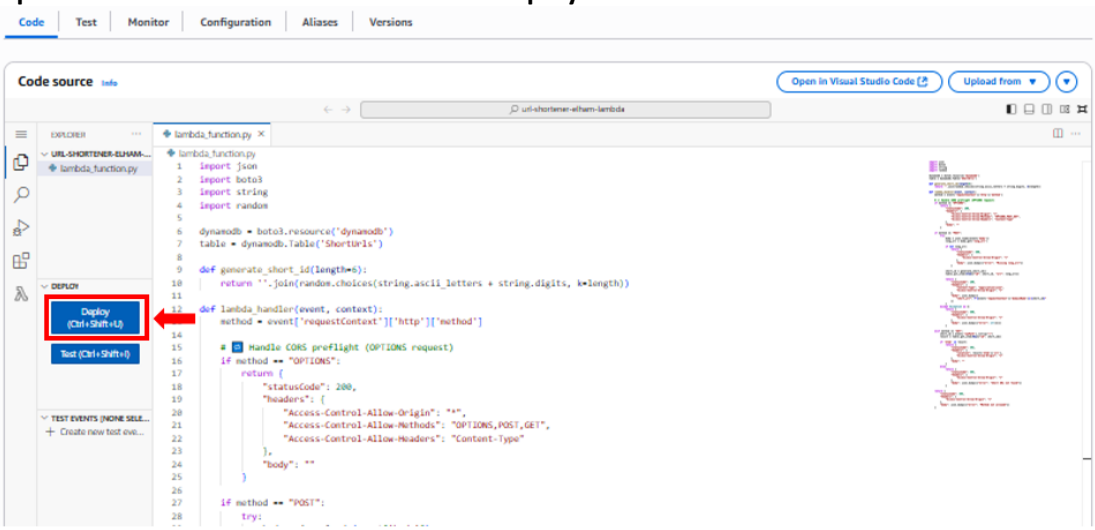
5. Execution role: **Use an existing role** → Select **labrole** , and then click **create**.



6. Successfully created " **url-shortener-elham-lambda**" function .

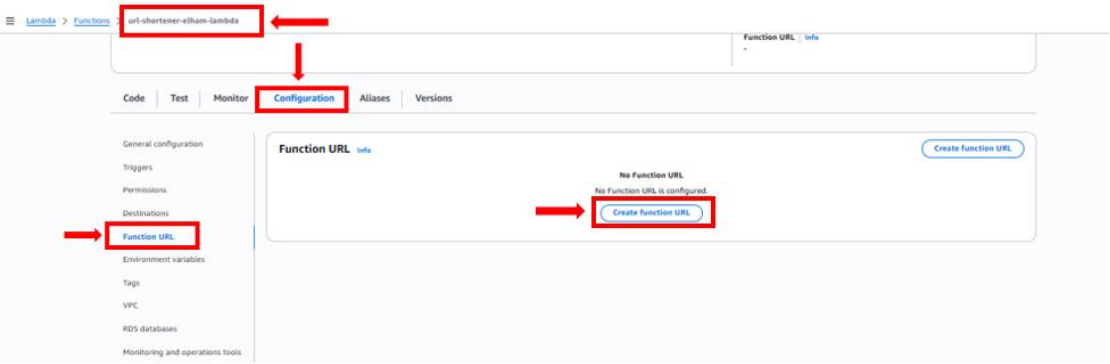


7. **Upload** the **code** into the editor and then **deploy** the function.

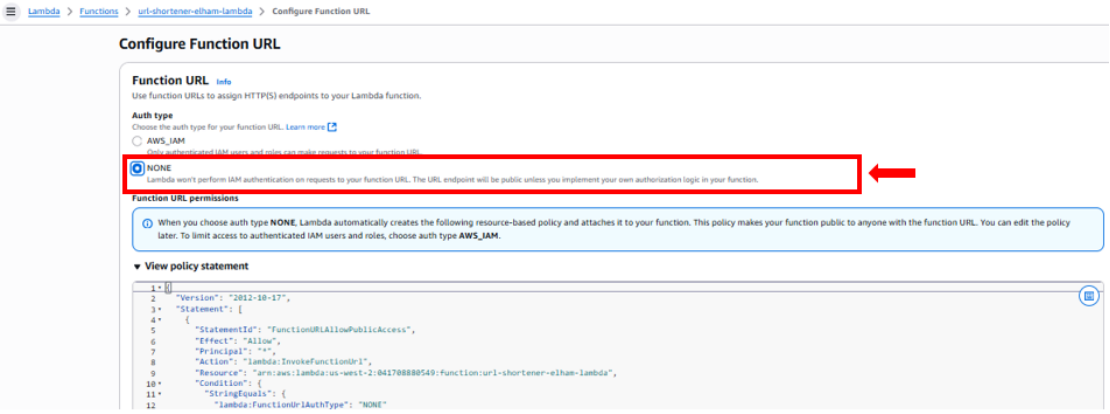


✅ Step 3: Enable Lambda Function URL

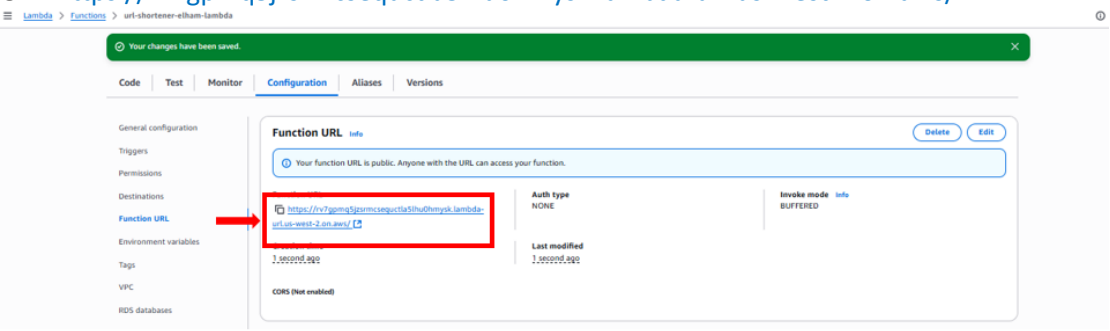
1. Go to your Lambda → Select **Configuration** → **Function URL** → Click **Create Function URL**.



2. Choose: **Auth type**: **NONE.**



3. Copy the **Function URL.**
URL:https://rv7gpmq5jzsrmcsequctla5lhu0hmysk.lambda-url.us-west-2.on.aws/



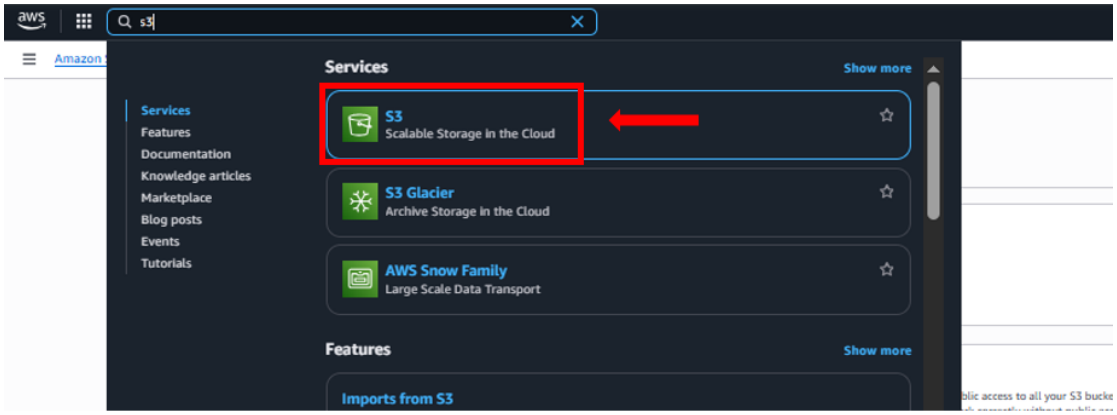
✅ Step 4: Connect Frontend to Lambda

1. In your **index.html** , update this line: const lambdaUrl = "https://your-lambda-url.on.aws/";

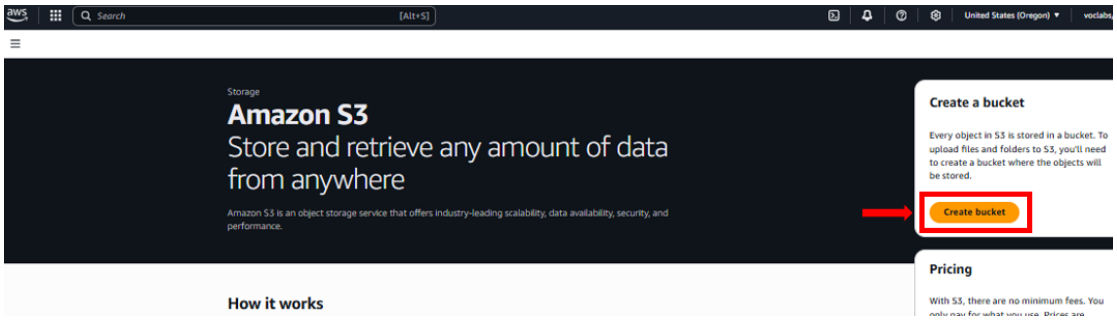
```
<script>
  const lambdaUrl = "https://rv7gpmq5jzsrmcsequctla5lhu0hmysk.lambda-url.us-west-2.on.aws/";
```

✅ Step 5: Create an S3 Bucket and Host the Frontend

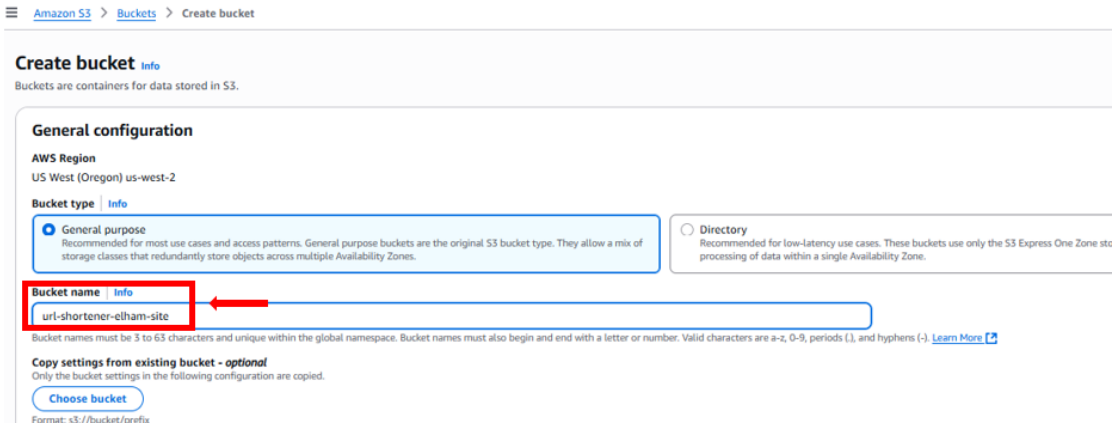
a. Go to the **S3 Console.**



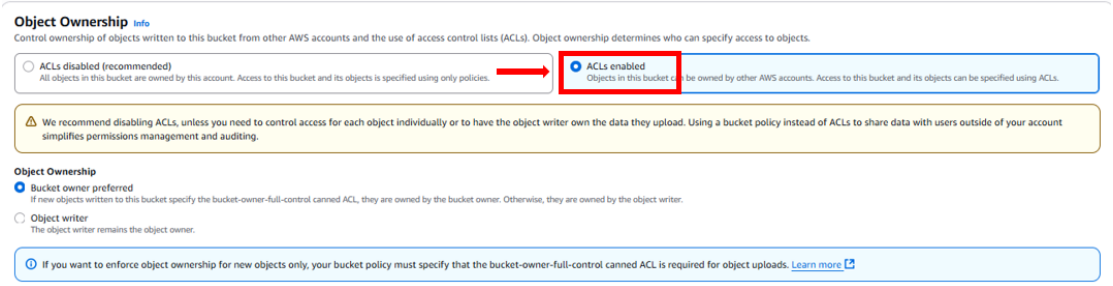
b. Click **Create bucket**.



c. Choose a name : "url-shortener-elham-site".



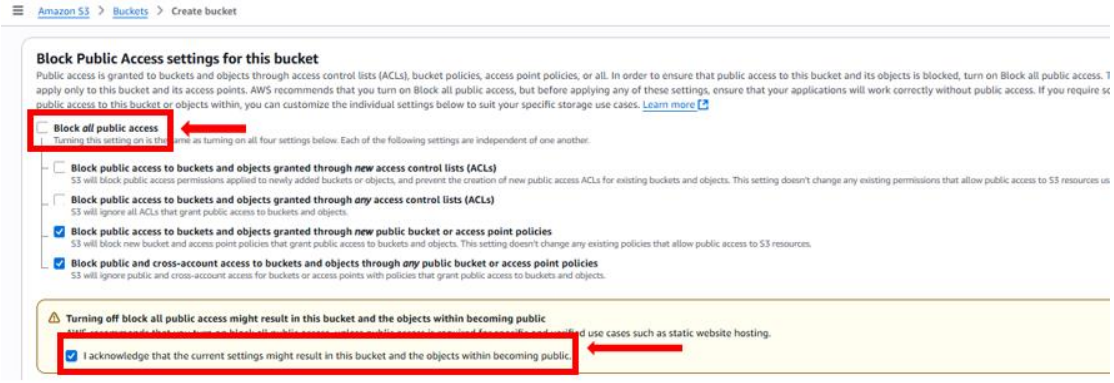
d. Under object ownership choose ACLs enabled.



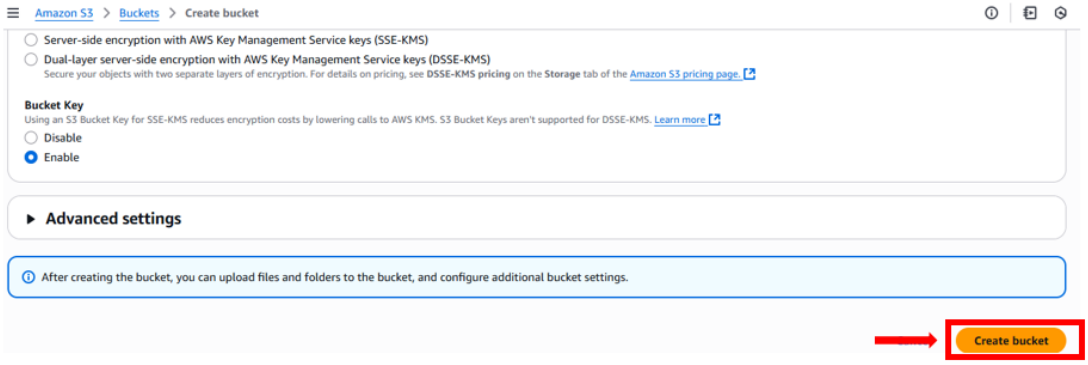
e. Under **Block public access (bucket settings)**, Uncheck **Block all public access** and then :

✓ Uncheck the **first & second option only**.

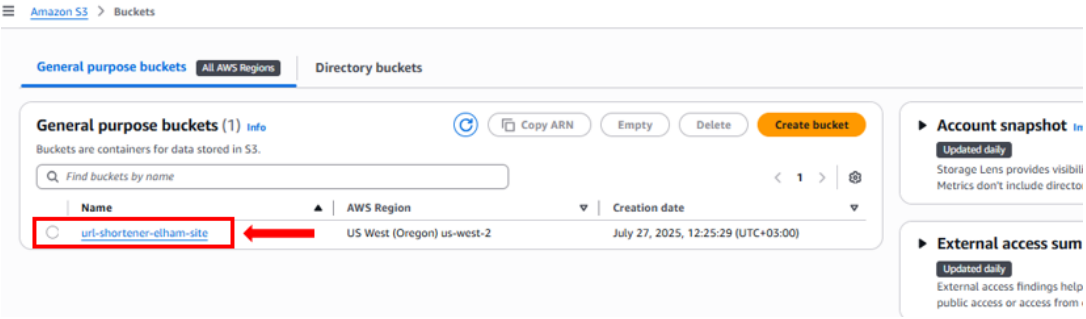
✓ Acknowledge the public warning.



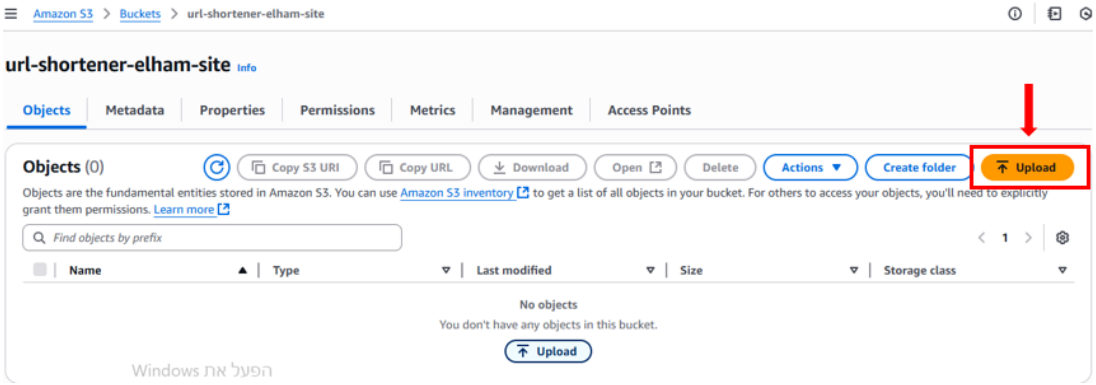
f. Click create.



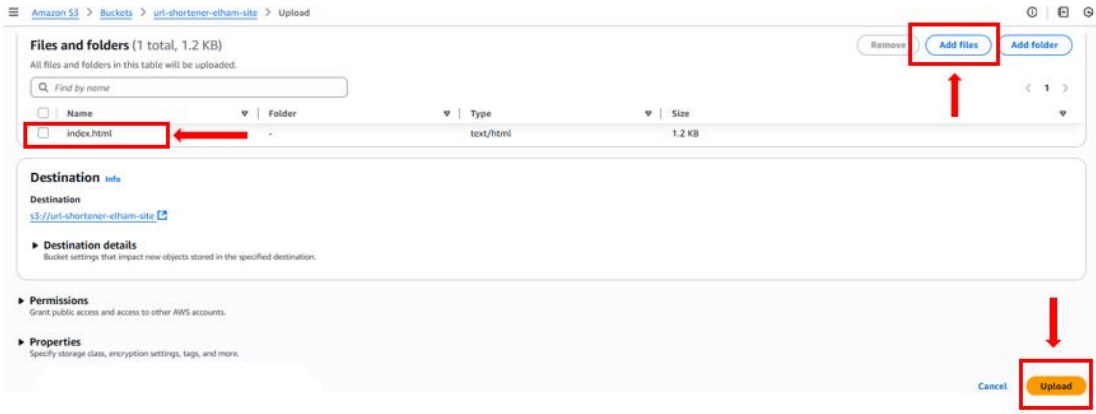
g. Successfully created "**url-shortner-elham-site**" bucket.



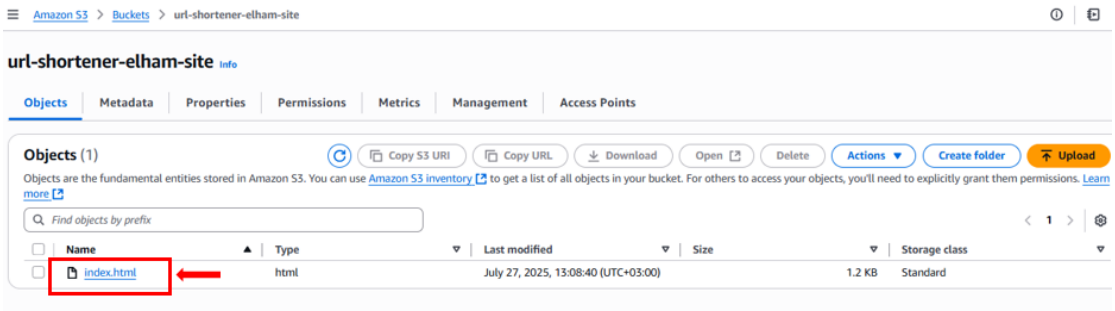
h. Click **upload** so you can add your files (**index.html , Style.css , background.jpg)** .



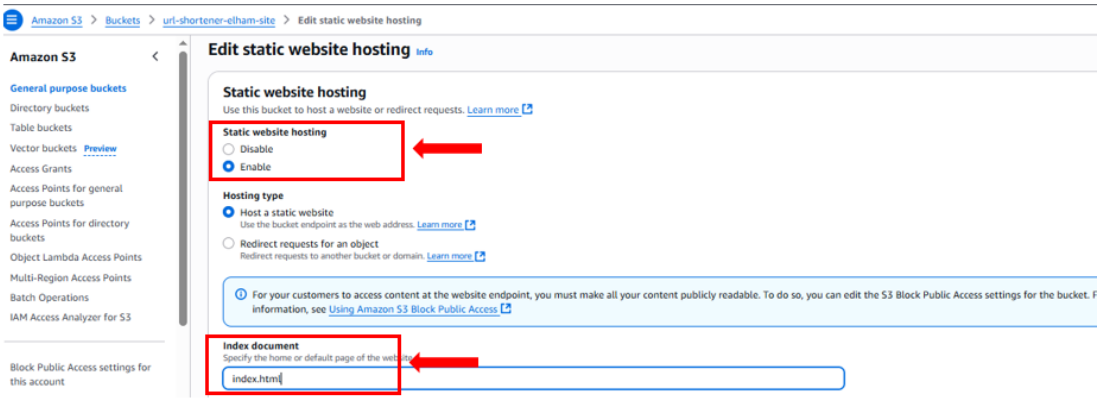
i. **Add** your file and then click **upload ,** example of adding **index.html** file .



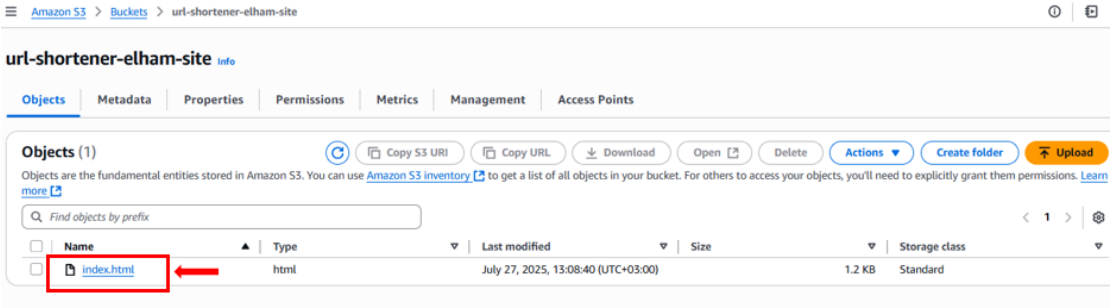
j. Successfully uploaded the **index.html** file , add the rest files in the same way.



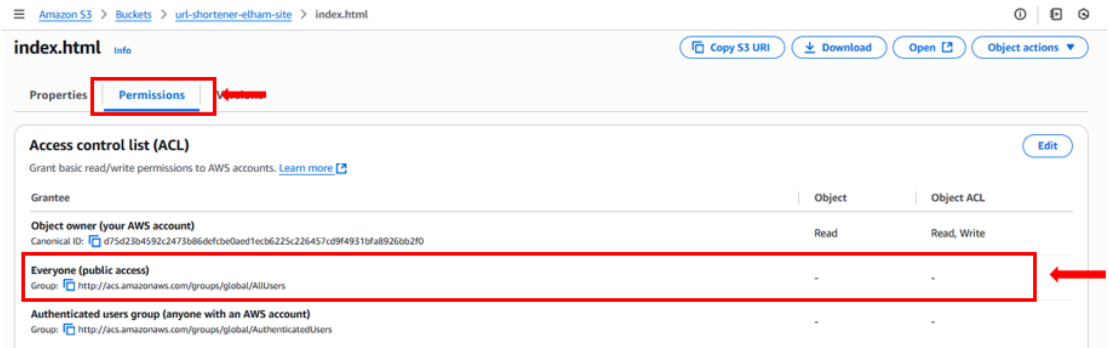
k. Enable **Static website** hosting in the **Properties** tab and Set index document to index.html.



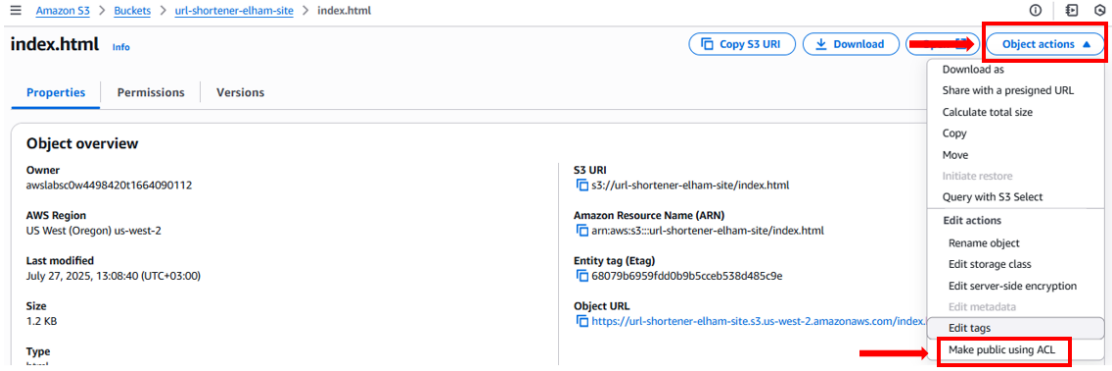
l. Check the file permissions (for example : **index.html** file), click on the file name .



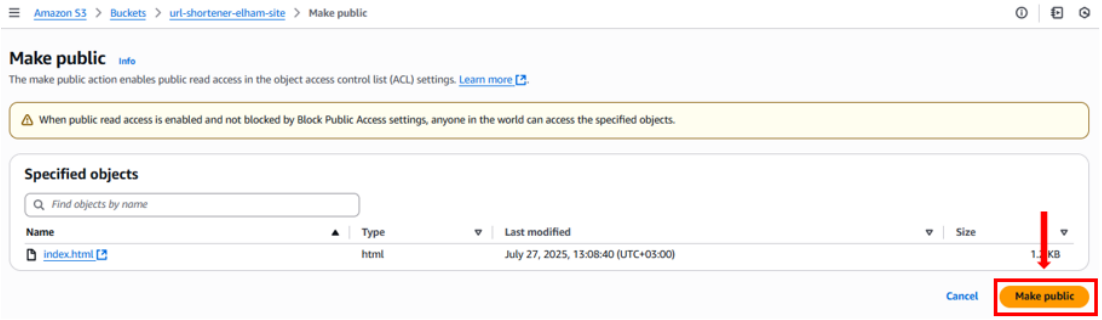
m. Click on **permissions**, you can see that there is no **public read access for everyone.**



n. Make the files **public** to allow public read access for everyone, click on **object actions** for every file and then choose "**Make public using ACL**" , for example : **index.html**.

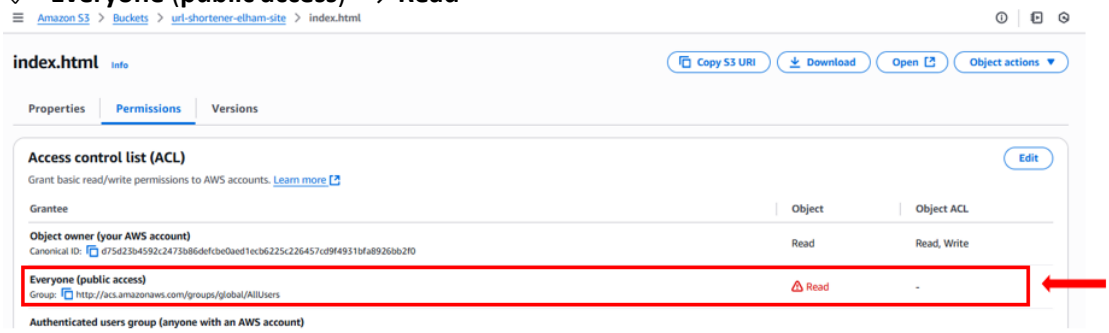


o. Click on **make public**.

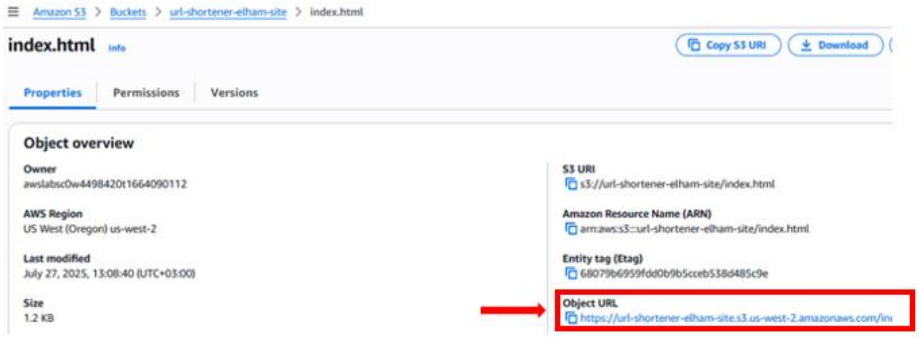


p. You can check all files **permissions** , for example : **index.html**.

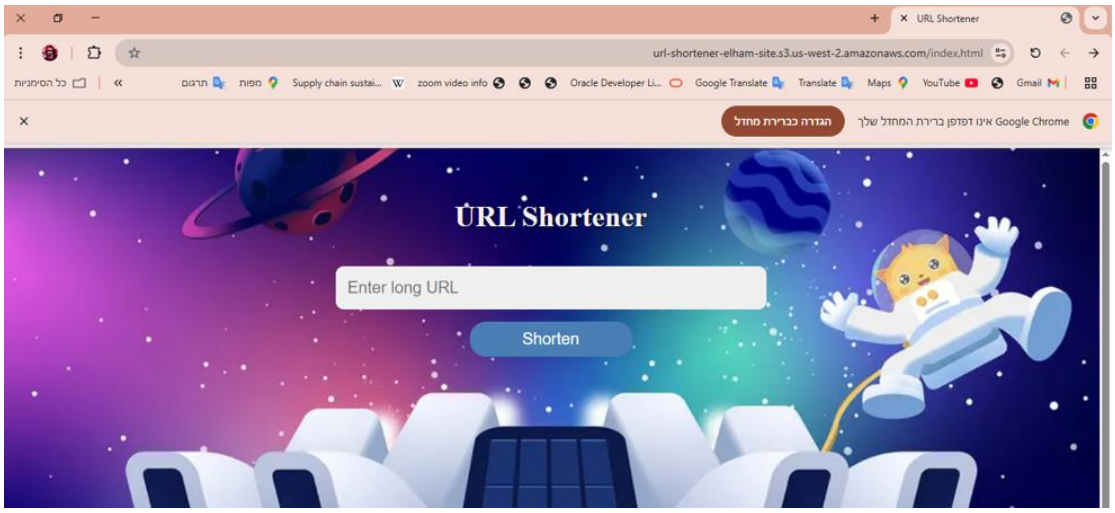
✅ "**Everyone** (**public access**)" → **Read**



q. Get the **website URL** : https://url-shortener-elham-site.s3.us-west-2.amazonaws.com/index.html .



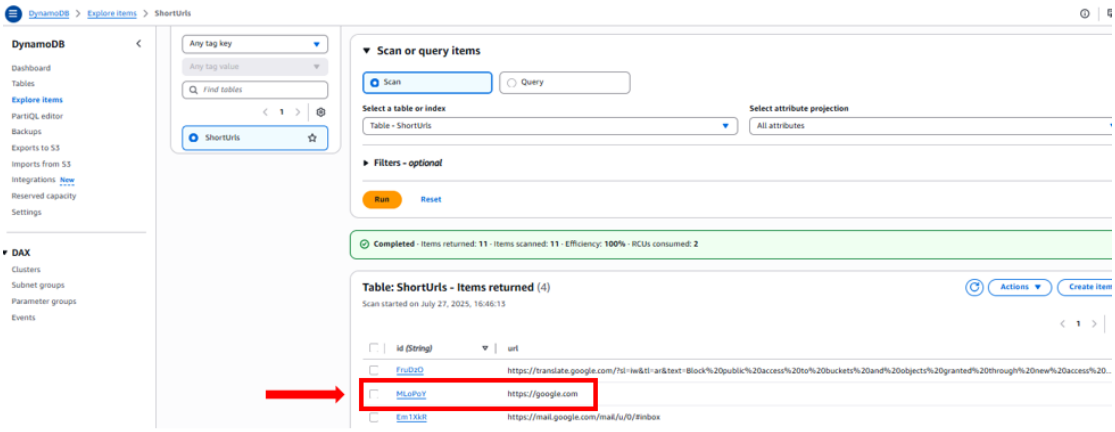
r. Open the **URL** in new tab.



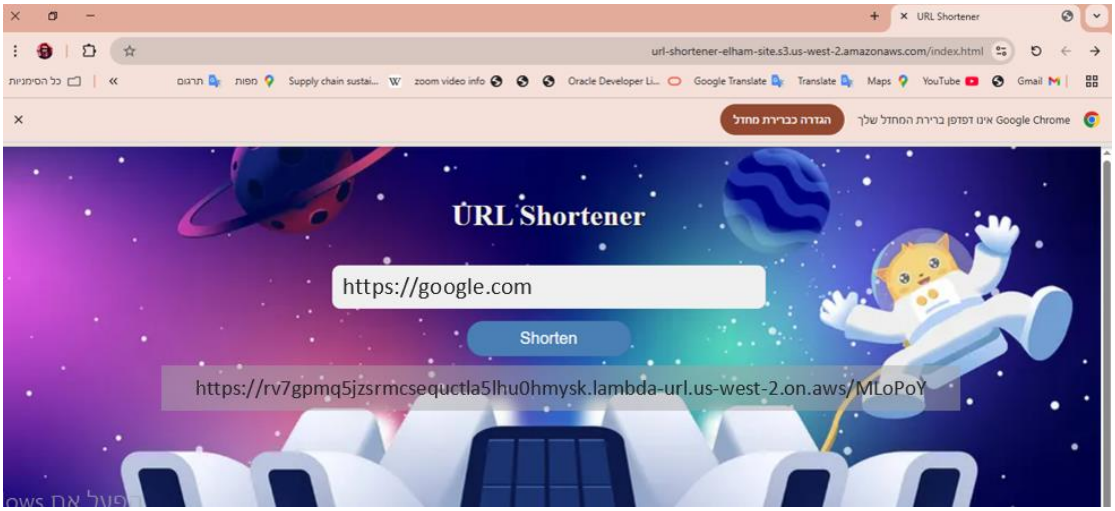
✅ Step 6: Test the Flow

1. Open your static site URL (from S3 hosting tab): https://url-shortener-elham-site.s3.us-west-2.amazonaws.com/index.html
2. Paste a long URL like https://google.com
3. The table stores mappings like:

id	url
MLoPoY	https://google.com



4. Click **Shorten**
5. You'll get a short URL like: https://rv7gpmq5jzsrmcsequctla5lhu0hmysk.lambda-url.us-west-2.on.aws/MLoPoY .



6. Click the short URL — it should redirect to the original one ✅