

Serverless URL Shortener using AWS Lambda, API Gateway, and DynamoDB

1. Project Overview

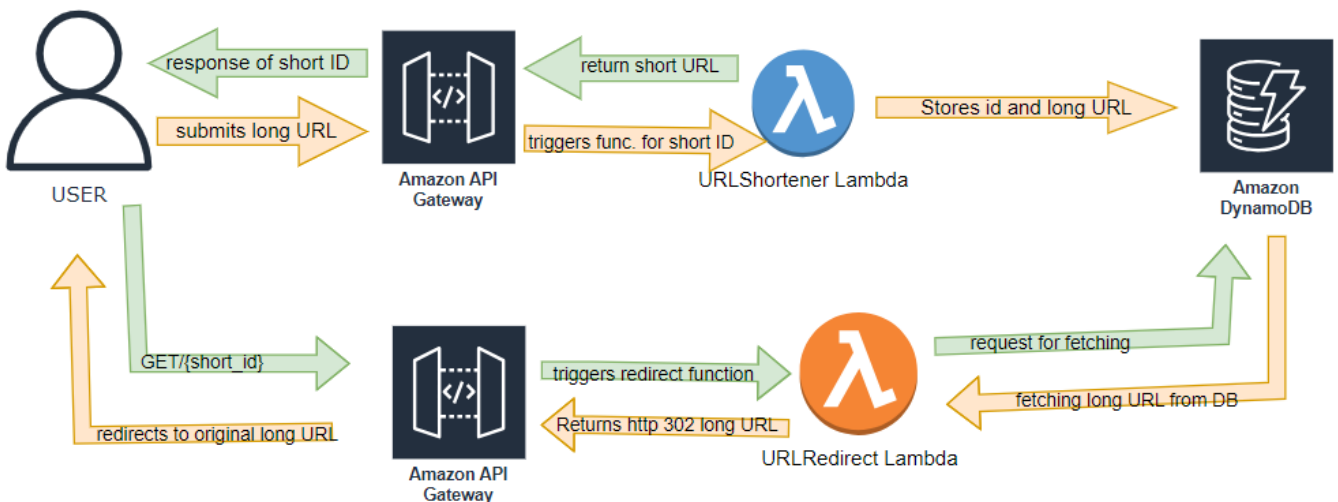
Create a serverless URL shortening application where users can:

- Submit a long URL and receive a shortened version.
- Use the short URL to redirect to the original long URL.

2. Architecture and Services Used

- **AWS API Gateway** — to expose RESTful endpoints for URL shortening and redirecting.
- **AWS Lambda** — functions to handle URL shortening logic and URL redirection.
- **DynamoDB** — to store the mapping between short IDs and long URLs.
- **IAM Roles and Policies** — to securely allow Lambda functions access to DynamoDB.
- **CloudWatch Logs** — to monitor and debug Lambda executions.

Serverless URL Shortener using AWS Lambda, API Gateway, and DynamoDB



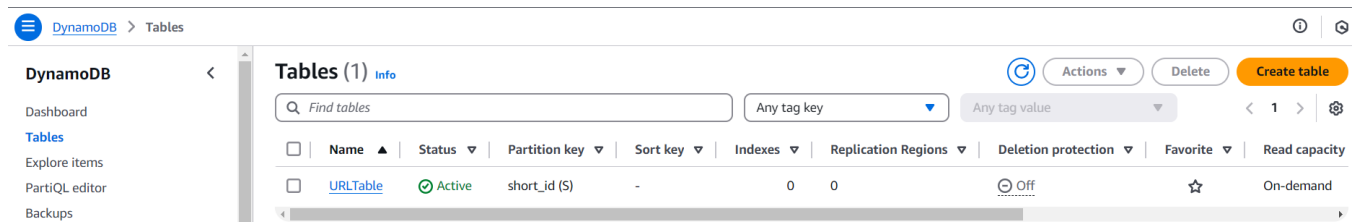
3. API Design

- **POST /shorten**
 - Request: JSON with a long_url field.
 - Response: JSON containing the generated short_url.
- **GET /{short_id}**
 - Request: Short ID path parameter.
 - Response: HTTP 302 redirect to the original long URL.

4. Implementation Steps

a. DynamoDB Table Setup

- Created a DynamoDB table named URLMapping.
- Primary key: short_id (string).
- Stores long_url against each short_id.



b. Lambda Function: URLShortener

- Receives POST request with long_url.
- Generates a unique short ID.
- Stores mapping (short_id → long_url) in DynamoDB.
- Returns JSON with the full short URL.

c. Lambda Function: URLRedirect

- Receives GET request with short_id.
- Queries DynamoDB for the original URL.
- Returns HTTP 302 redirect with Location header pointing to the long_url.

☰ [Lambda](#) > Functions

Functions (2) Last fetched 0 seconds ago Actions Create function

Q Filter by attributes or search by keyword

<input type="checkbox"/>	Function name	Description	Package type	Runtime	Last modified
<input type="checkbox"/>	URLRedirect	-	Zip	Python 3.12	45 minutes ago
<input type="checkbox"/>	URLShortener	-	Zip	Python 3.12	50 minutes ago

d. API Gateway Configuration

- Created a REST API.
- Configured POST /shorten method integrated with the URLShortener Lambda.
- Configured GET /{short_id} method integrated with the URLRedirect Lambda.
- Set up method request and integration request mappings.
- Enabled logging and tracing for debugging.

Resources API actions Deploy API

Create resource

/

/shorten

Resource details

Path /shorten

Resource ID zo1zs4

Methods (0) Delete Create method

Method type	Integration type	Authorization	API key
No methods			

No methods defined.

Resources API actions Deploy API

Create resource

/

/shorten

POST

/shorten - POST - Method execution Update documentation Delete

ARN [arn:aws:execute-api:us-east-1:123456789012:zo1zs4/*/*/POST/shorten](#)

Resource ID zo1zs4

Client

Method request

Integration request

Method response

Integration response

Proxy integration

Lambda integration

Method request Integration request Integration response Method response Test

API actions ▼ **Deploy API**

Deploy API

Create or select a stage where your API will be deployed. You can use the deployment history to revert or change the active deployment for a stage. [Learn more](#)

Stage

New stage

Stage name

dev

A new stage will be created with the default settings. Edit your stage settings on the **Stage** page.

Deployment description

Cancel

Deploy


API actions

The diagram illustrates the execution flow for the `GET /short_id` endpoint. It shows a sequence of steps: a **Client** (represented by a computer icon) sends a **Method request** to the **Integration request** (represented by a Lambda icon). The **Integration request** then triggers the **Integration response** (labeled **Proxy integration**), which returns a **Method response** to the **Client**. The flow is indicated by arrows connecting the components in a cycle.

```
graph LR; Client[Client] --> MethodRequest[Method request]; MethodRequest --> IntegrationRequest[Integration request]; IntegrationRequest --> IntegrationResponse[Integration response  
Proxy integration]; IntegrationResponse --> MethodResponse[Method response]; MethodResponse --> Client;
```

5. IAM Roles and Permissions

- Created IAM roles granting Lambda functions permission to read/write the DynamoDB table.
- Assigned roles to respective Lambda functions.

Roles (13) Info					Delete
An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.					
<input type="text" value="Search"/>					
<input type="checkbox"/>	Role name	▲	Trusted entities	Last activity	
<input type="checkbox"/>	s3crr_role_for_firstbucketinohiooo		AWS Service: s3	8 days ago	
<input type="checkbox"/>	URLRedirect-role-t82nvl7s		AWS Service: lambda	-	
<input type="checkbox"/>	URLShortener-role-oas1ecnf		AWS Service: lambda	-	

6. Testing

- Used the **API Gateway Console** to test both API methods directly:
 - **POST /shorten**

Input: JSON body with long_url, e.g.:

```
{
  "long_url": "https://www.example.com"
}
```

Output: JSON response with the generated short URL, for example:

```
{
```

```
  "short_url": "https://<api-id>.execute-api.<region>.amazonaws.com/dev/4ckeuj"
}
```

The screenshot shows the AWS API Gateway console for the URLShortenerAPI (cne66c81k9). The left sidebar lists various API Gateway features, including APIs, Custom domain names, Domain name access associations, VPC links, and API settings. The main panel displays the API path /{short_id} with GET and POST methods. The POST method is selected, and the request body is shown as a JSON object: {\"long_url\": \"https://www.example.com\"}. The test results for the POST method are displayed, showing a status of 200 and a latency of 951 ms. The response body is {\"short_url\": \"https://<api-id>.execute-api.<region>.amazonaws.com/dev/4ckeuj\"}.

○ GET /{short_id}

- Tested by entering a short ID (e.g., 4ckeuj) in the path parameter field in the API Gateway Console test.
- The API Gateway skipped authorization and directly invoked the Lambda function.
- Response returned HTTP status **302** with a Location header redirecting to the original URL (https://www.example.com).
- Example response headers from the test:
- Status: 302
- Location: <https://www.example.com>

API Gateway > APIs > Resources - URLShortenerAPI (cne6Gc81k9)

API Gateway

- APIs
- Custom domain names
- Domain name access associations
- VPC links

▼ API: URLShortenerAPI

- Resources
- Stages
- Authorizers
- Gateway responses
- Models
- Resource policy
- Documentation
- Dashboard
- API settings

Usage plans

API keys

Client certificates

Settings

Path

short_id

4ckeuj

Query strings

param1=value1¶m2=value2

Headers

header1:value1
header2:value2

Client certificate

No client certificates have been generated.

Test

ⓘ /{short_id} - GET method test results

Request	Latency ms	Status
/4ckeuj	1031	302

Response body

No data

Response headers

```
{
  "Location": "https://www.example.com",
  "X-Amzn-Trace-Id": "Root=1-684a01e5-898757f8119e800c03fc806d;Parent=0c1bd0efd5738861;Sampled=0;Lineage=1:3ee0ed4b:0"
}
```

Logs

Verified logs in CloudWatch confirming Lambda executions and integration latencies.

CloudWatch > Log groups

CloudWatch

Log groups (4)

By default, we only load up to 10000 log groups.

Filter log groups or try pattern search

Exact match

Log group	Log class	Anomaly d...	Data
/aws/lambda/URLRedirect	Standard	Configure	-
/aws/lambda/URLShortener	Standard	Configure	-
elasticenginelogs	Standard	Configure	-
elasticlogs	Standard	Configure	-

CloudWatch > Log groups > /aws/lambda/URLRedirect

CloudWatch <

Log group details

Log class: Standard

ARN: arn:aws:logs:us-east-1:418295712814:log-group:/aws/lambda/URLRedirect:*

Creation time: 24 minutes ago

Retention: Never expire

Stored bytes: -

Metric filters: 0

Subscription filters: 0

Contributor Insights rules: -

KMS key ID: -

Anomaly detection: Configure

Data protection: -

Sensitive data count: -

Field indexes: Configure

Transformer: Configure

Log streams (3)

Filter log streams or try prefix search

Exact match Show expired Info

Log stream	Last event time
2025/06/11/[\${LATEST}]8b26d54b6154429c8048f8e3b7382484	2025-06-11 22:23:34 (UTC)
2025/06/11/[\${LATEST}]b35a3551cd9440a796afdd0f05529ea1	2025-06-11 22:13:11 (UTC)
2025/06/11/[\${LATEST}]ef9ac9168e334c93848d619b0c6c3d0a	2025-06-11 22:04:58 (UTC)

CloudWatch > Log groups > /aws/lambda/URLShortener

CloudWatch <

Log group details

Log class: Standard

ARN: arn:aws:logs:us-east-1:418295712814:log-group:/aws/lambda/URLShortener:*

Creation time: 27 minutes ago

Retention: Never expire

Stored bytes: -

Metric filters: 0

Subscription filters: 0

Contributor Insights rules: -

KMS key ID: -

Anomaly detection: Configure

Data protection: -

Sensitive data count: -

Field indexes: Configure

Transformer: Configure

Log streams (3)

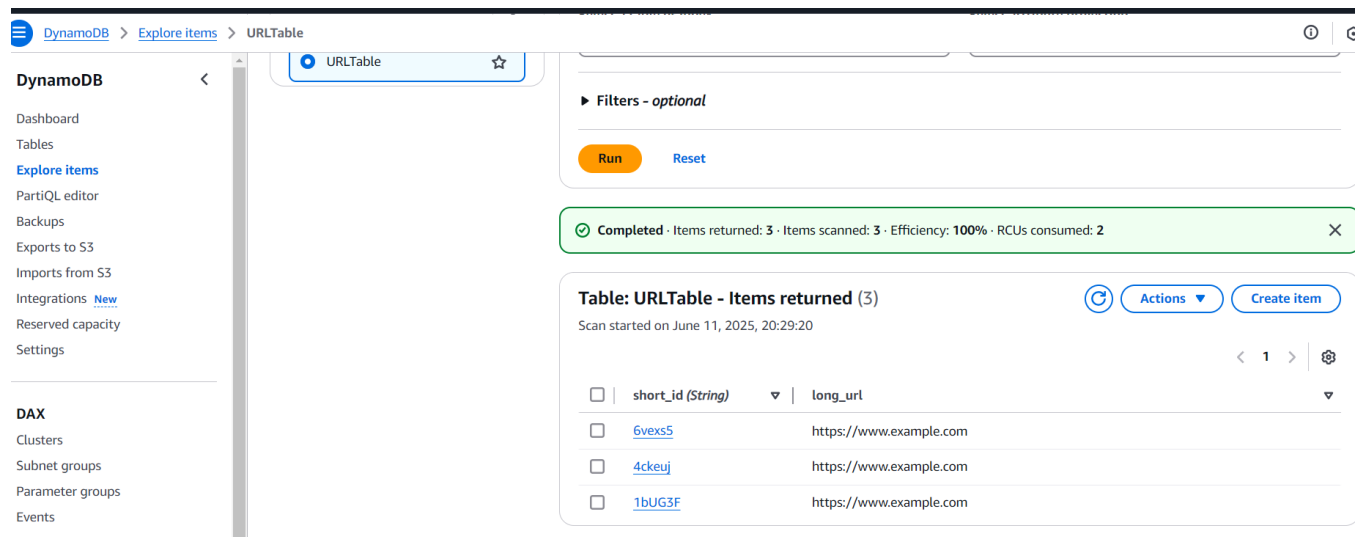
Filter log streams or try prefix search

Exact match Show expired Info

Log stream	Last event time
2025/06/11/[\${LATEST}]0172fba76916448d86a493209f11361a	2025-06-11 22:21:07 (UTC)
2025/06/11/[\${LATEST}]f12d65953f5c48a3949e97d575d0dfba	2025-06-11 22:11:50 (UTC)
2025/06/11/[\${LATEST}]d65a205f29448c69da1c25a1a2ccfad	2025-06-11 22:02:37 (UTC)

- Confirmed end-to-end functionality: submitting a long URL produces a short URL, and using the short URL correctly redirects to the original site.

7. DynamoDB key-value storage:



8. Challenge Faced & Resolution

While configuring the **GET method** in API Gateway to retrieve the original long URL using a dynamic path parameter (e.g., /short_id), I encountered a key challenge: entering {short_id} as the **resource path** in the console was mistakenly interpreted as a static string instead of a dynamic path variable. This resulted in API Gateway treating /short_id literally, rather than allowing unique IDs to be passed dynamically. After several failed tests and configuration attempts, the issue was resolved with the help of AWS CloudShell. By using a command like:

```
aws apigateway create-resource --rest-api-id cne66c81k9 --parent-id yob4bjpcsj --path-part '{short_id}' --region us-east-1
```

creating GET method :

```
aws apigateway put-method --rest-api-id cne66c81k9 --resource-id abc123 --http-method GET --authorization-type NONE --region us-east-1
```

Integrate the GET method with Lambda function

```
aws apigateway put-integration --rest-api-id cne66c81k9 --resource-id abc123 --http-method GET --type AWS_PROXY --integration-http-method POST --uri arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/<lambda-arn>/invocations --region us-east-1
```

Add the permission and deploy:

```
aws apigateway create-deployment --rest-api-id cne66c81k9 --stage-name dev --region us-east-1
```

I confirmed that the API Gateway could correctly interpret and route dynamic paths once the resource was properly configured. This validation through CloudShell clarified the importance of precise

resource path syntax and dynamic variable setup within API Gateway. The method began working as intended once the path parameter was correctly defined and tested outside the console interface.

9. Conclusion

This project demonstrates building a simple, scalable, serverless URL shortener using AWS managed services. It provides hands-on experience with API Gateway, Lambda, DynamoDB, and CloudWatch, showcasing how to design REST APIs and implement redirect logic.