

1. Building a Serverless Web Application

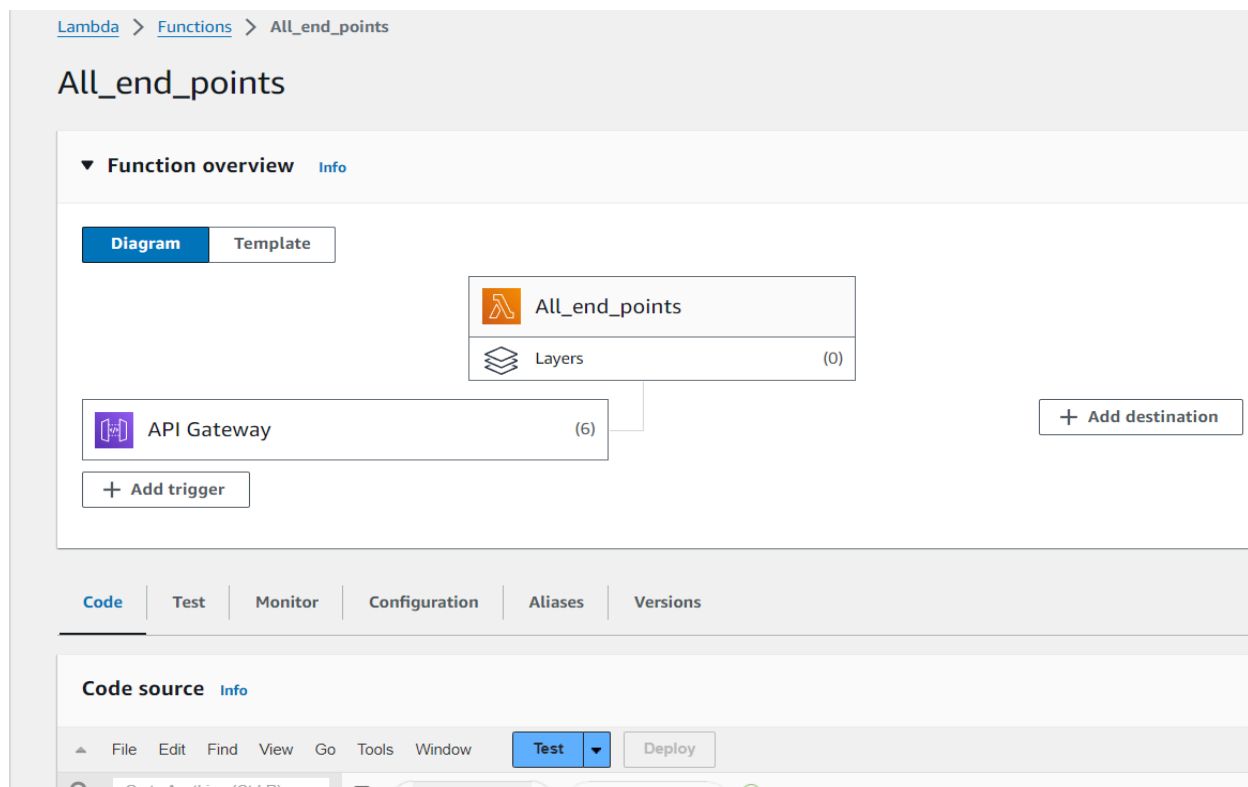
Objective: Create a serverless web application using AWS Lambda, API Gateway, S3, and DynamoDB.

Approach:

- **Set Up Backend:** Create Lambda functions to handle backend logic. These functions will interact with a DynamoDB table for data storage.
- **API Gateway:** Set up API Gateway to create RESTful endpoints that trigger the Lambda functions.
- **Frontend Hosting:** Host a static website on S3 that interacts with the backend via API Gateway.
- **Integration:** Ensure that the frontend can successfully send requests to the backend and display responses.

Goal: Understand the basics of building and connecting serverless backend services with a static frontend, enabling a fully serverless web application.

Lambda function structure:



Same structure of serverless 2 is used.

Also the Api gateway integration is shown in serverless lab 2.

Enabling cors for api gateway:

API Gateway > APIs > Resources - All_end_points-API (x2dnb3xqk5) > Enable CORS

Enable CORS

CORS settings [Info](#)

To allow requests from scripts running in the browser, configure cross-origin resource sharing (CORS) for your API.

Gateway responses
API Gateway will configure CORS for the selected gateway responses.

☐ Default 4XX

☐ Default 5XX

Methods

☒ DELETE

☒ GET

☒ OPTIONS

☒ POST

☒ PUT

Access-Control-Allow-Methods
DELETE, GET, HEAD, OPTIONS, PATCH, POST, PUT

Access-Control-Allow-Headers
API Gateway will configure CORS for the selected gateway responses.

Access-Control-Allow-Origin
Enter an origin that can access the resource. Use a wildcard "*" to allow any origin to access the resource.

Create a S3 bucket to add the website files:

Amazon S3 > Buckets

▼ **Account snapshot**

Last updated: Feb 15, 2024 by Storage Lens. Metrics are generated every 24 hours. Metrics don't include directory buckets. [Learn more](#)

Total storage	Object count	Average object size
2.3 KB	9	260.1 B

General purpose buckets | **Directory buckets**

General purpose buckets (3) [Info](#)

Buckets are containers for data stored in S3. [Learn more](#)

	Name ▲	AWS Region ▼	Access ▼	Creation date
<input type="radio"/>	myfirstbucket-1313	US East (N. Virginia) us-east-1	Objects can be public	January 22, 2024,
<input type="radio"/>	static-bucket36	US East (N. Virginia) us-east-1	Objects can be public	February 12, 2024
<input type="radio"/>	web13staticlab2	US East (N. Virginia) us-east-1	Public	February 16, 2024

Add the html and js file to s3:

[Amazon S3](#) > [Buckets](#) > [web13staticlab2](#) > Upload

Upload [Info](#)

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.

Files and folders (2 Total, 8.0 KB) [Remove](#) [Add files](#) [Add folder](#)

All files and folders in this table will be uploaded.

Find by name < 1 >

<input type="checkbox"/>	Name	Folder	Type
<input type="checkbox"/>	index.html	-	text/html
<input type="checkbox"/>	script.js	-	text/javascript






Destination [Info](#)

All files in S3:

Objects (5) [Info](#) [Refresh](#) [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#)

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

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size
<input type="checkbox"/>	 index.html	html	February 17, 2024, 13:44:27 (UTC+05:45)	
<input type="checkbox"/>	 index1.html	html	February 16, 2024, 22:35:29 (UTC+05:45)	
<input type="checkbox"/>	 ppss.jpg	jpg	February 16, 2024, 22:36:43 (UTC+05:45)	
<input type="checkbox"/>	 script.js	js	February 17, 2024, 13:22:10 (UTC+05:45)	
<input type="checkbox"/>	 style.css	css	February 17, 2024, 13:44:28 (UTC+05:45)	

Enable static hosting:

Static website hosting

Use this bucket to host a website or redirect requests. [Learn more](#)

Static website hosting

☐ Disable
☒ Enable

Hosting type

☒ Host a static website
 Use the bucket endpoint as the web address. [Learn more](#)

☐ Redirect requests for an object
 Redirect requests to another bucket or domain. [Learn more](#)

For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see [Using Amazon S3 Block Public Access](#)

Index document

Specify the home or default page of the website.

Error document - *optional*

This is returned when an error occurs.

Now navigate to the provided static website URL:

Static website hosting

Use this bucket to host a website or redirect requests. [Learn more](#)

Static website hosting

Enabled

Hosting type

Bucket hosting

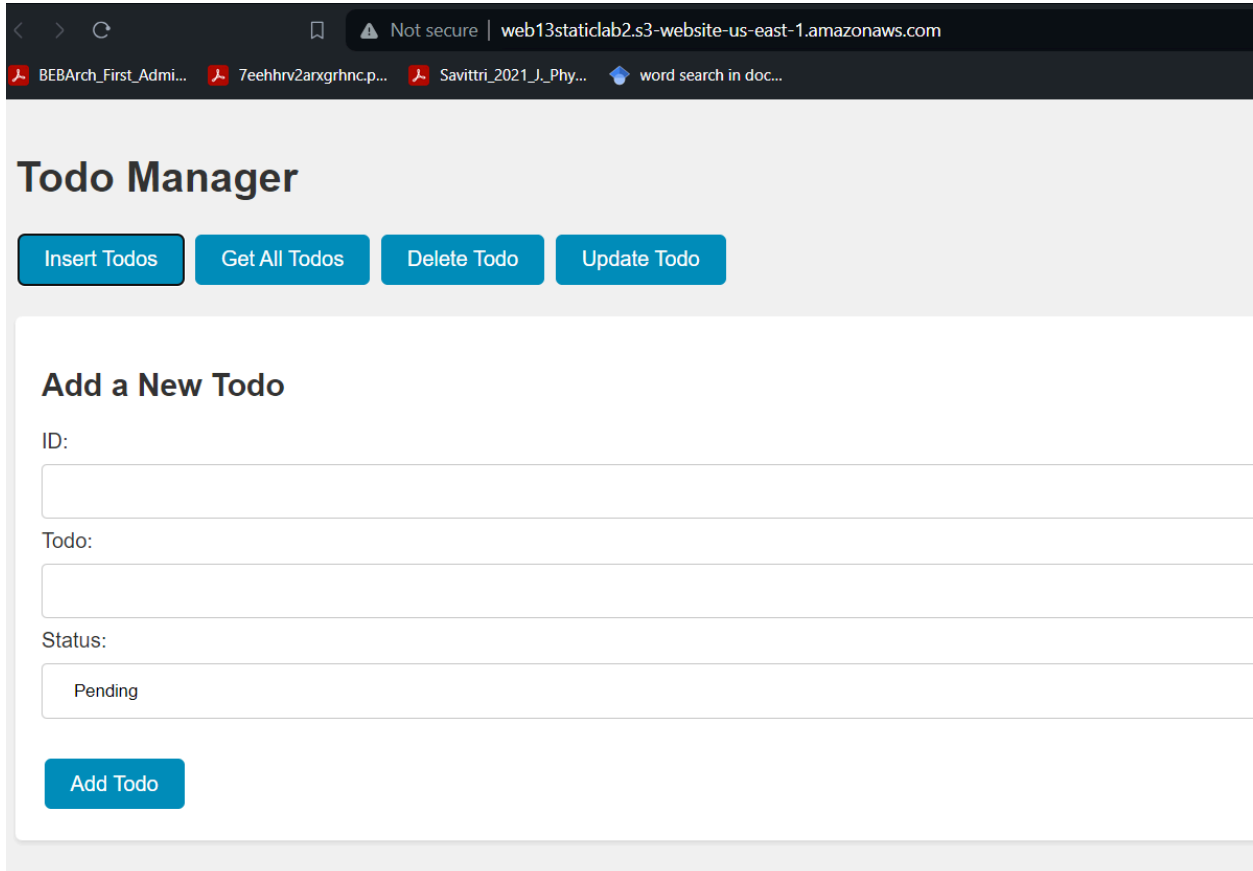
Bucket website endpoint

When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. [Learn more](#)

<http://web13staticlab2.s3-website-us-east-1.amazonaws.com>

Checking the application functionalities:

Basic UI for ToDo application:



The screenshot shows a web browser window with the address bar displaying 'web13staticlab2.s3-website-us-east-1.amazonaws.com'. The browser tabs include 'BEBArch_First_Admi...', '7eehhrv2arxgrhnc.p...', 'Savittri_2021_J_Phy...', and 'word search in doc...'. The main content area is titled 'Todo Manager' and features four blue buttons: 'Insert Todos', 'Get All Todos', 'Delete Todo', and 'Update Todo'. Below these buttons is a section titled 'Add a New Todo' with three input fields: 'ID:', 'Todo:', and 'Status:'. The 'Status:' field is currently set to 'Pending'. At the bottom of this section is a blue 'Add Todo' button.

Todo Manager

Insert Todos Get All Todos Delete Todo Update Todo

Add a New Todo

ID:

Todo:

Status:

Add Todo

Adding the todos:

Insert Todos

Get All Todos

Delete Todo

Update Todo

Add a New Todo

ID:

Todo:

Status:

Add Todo

word sea

web13staticlab2.s3-website-us-east-1.amazonaws.com says
Task submitted successfully!

OK

date Todo

View all Todos:

Todo Manager

[Insert Todos](#)
[Get All Todos](#)
[Delete Todo](#)
[Update Todo](#)

- ID: 8, Todo: Add Global page88, Status: Done
- ID: 1, Todo: added, Status: Done
- ID: 4, Todo: Add Global page, Status: Pending
- ID: 44, Todo: Add Global page44, Status: Pending
- ID: 23, Todo: final code push to Github, Status: Done
- ID: 7, Todo: Add Global page 777, Status: Pending
- ID: 3, Todo: Add Home page, Status: Pending

Also we can visualize in dynamoDb table as:

Completed. Read capacity units consumed: 0.5

Items returned (7)					Act
<input type="checkbox"/>	id (String)	▼	status	▼	todo
<input type="checkbox"/>	8		Done		Add Global page88
<input type="checkbox"/>	1		Done		added
<input type="checkbox"/>	4		Pending		Add Global page
<input type="checkbox"/>	44		Pending		Add Global page44
<input type="checkbox"/>	23		Done		final code push to Github
<input type="checkbox"/>	7		Pending		Add Global page 777
<input type="checkbox"/>	3		Pending		Add Home page

Delete todo functionality:

Todo Manager

Insert TodosGet All TodosDelete TodoUpdate Todo

Delete a Todo

ID of the todo to delete:

Delete Todo

web13staticlab2.s3-website-us-east-1.amazonaws.com says

Task deleted successfully!

OK

All todos after deleting todo with id=1:

Todo Manager

Insert TodosGet All TodosDelete TodoUpdate Todo

ID: 8, Todo: Add Global page88, Status: Done

ID: 4, Todo: Add Global page, Status: Pending

ID: 44, Todo: Add Global page44, Status: Pending

ID: 23, Todo: final code push to Github, Status: Done

ID: 7, Todo: Add Global page 777, Status: Pending

ID: 3, Todo: Add Home page, Status: Pending

Update functionality:

Todo Manager

Insert TodosGet All TodosDelete TodoUpdate Todo

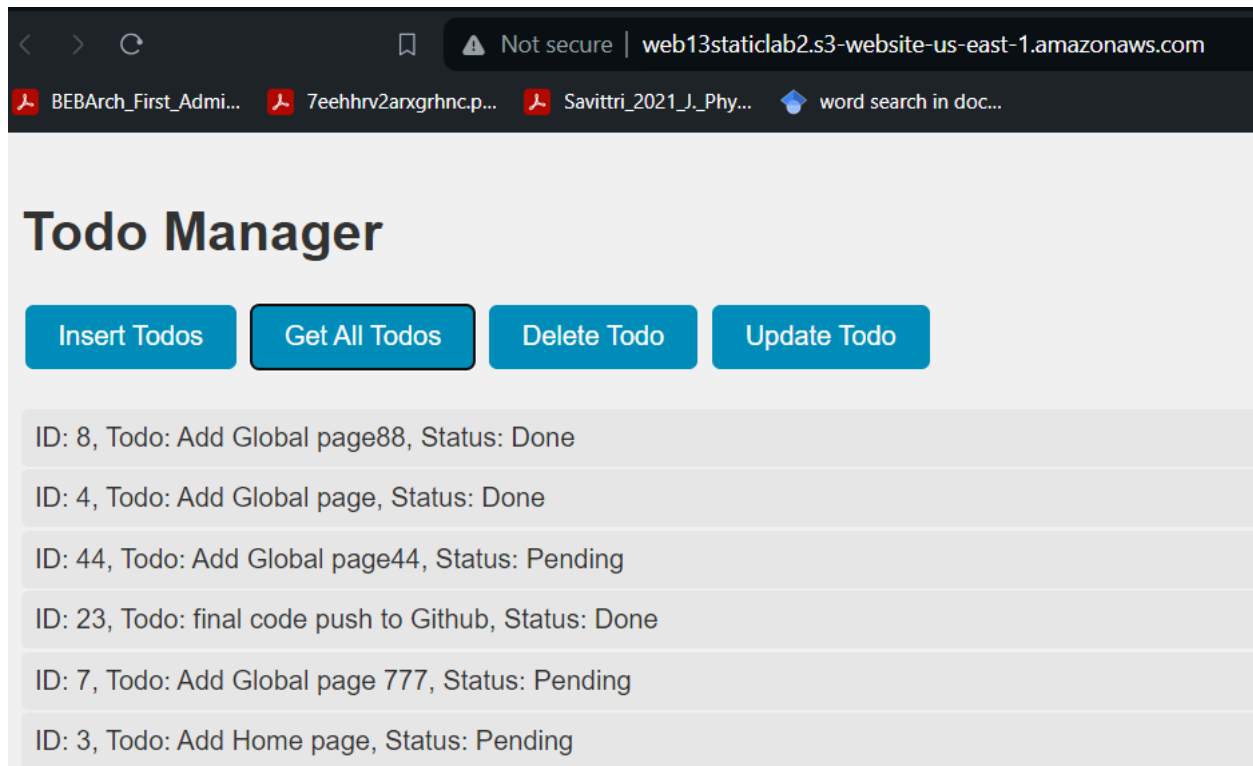
Update Todo Status

ID of the todo to update:

New Status:

Update Todo

After update:



This concludes the serverless lab 1 where we create lambda function for backend, dynamodb for database, api integration for api interaction and S3 static website hosting.