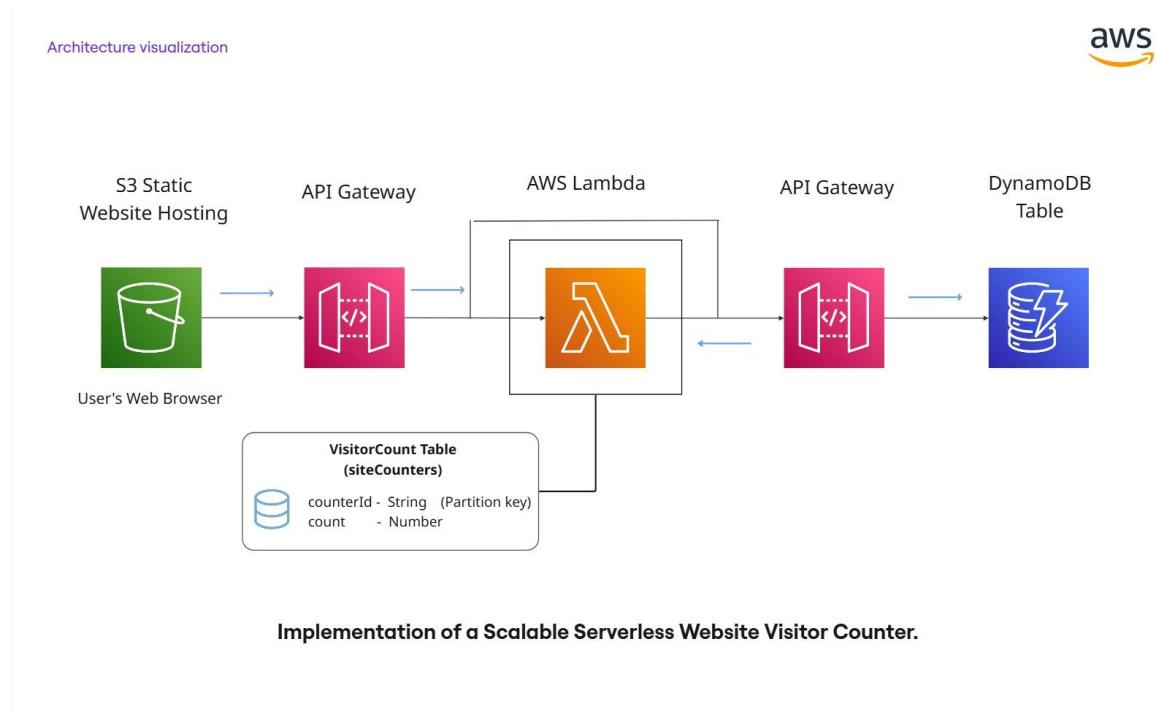


Implementation of a Scalable Serverless Website Visitor Counter.

AWS SERVICES USED:

1. AWS Lambda
2. API Gateway
3. DynamoDB
4. S3 (Simple Storage Service).

AWS ARCHITECTURAL DIAGRAM:



- This diagram shows a **serverless website visitor counter architecture** using AWS services.
- A static site on S3 calls API Gateway, which triggers a Lambda function that reads and updates visitor counts stored in DynamoDB.

OVERVIEW (ONE-LINE)

User visits site → browser `fetch()` hits API Gateway → API Gateway calls Lambda → Lambda atomically 'ADD's 1 to a DynamoDB item and returns the new count → browser displays it.

→ Here, we will see step by step process of how this architecture works in detail.

AWS IAM:

- IAM - Identity and Access Management is a service that lets you securely manage access to AWS resources.
- It enables you to control who can do what in your AWS account through users, roles, and policies.

The screenshot shows the AWS IAM Dashboard. On the left, there's a sidebar with 'Identity and Access Management (IAM)' selected. Under 'Access management', 'Policies' is highlighted. In the center, 'Security recommendations' include 'Root user has MFA' and 'Root user has no active access keys'. 'IAM resources' show 0 User groups, 0 Users, 9 Roles, 7 Policies, and 0 Identity providers. On the right, the 'AWS Account' section shows the Account ID (820314685364) and a 'Sign-in URL for IAM users in this account' (https://820314685364.signin.aws.amazon.com/console). A 'Quick Links' section includes 'My security credentials' and a note about managing access keys, MFA, and other credentials.

Step:1 - Create IAM role for Lambda:

- Minimal policy — allows Lambda to update the table and write logs.
 - IAM policy JSON (attach to role).
1. Choose trusted entity

Select: **AWS service**

Choose: **Lambda** and Click Next

The screenshot shows the 'Create role' wizard, Step 1: Select trusted entity. It lists four options: 'AWS service' (selected), 'AWS account', 'Web identity', and 'SAML 2.0 federation'. Below this, the 'Use case' section allows selecting a service or use case, with 'Lambda' chosen. At the bottom, there's a note about choosing a use case for the specified service.

2. Attach permissions: - Search for these policies

AmazonDynamoDBFullAccess - Lambda can update the counter

CloudWatchLogsFullAccess - Lambda can write logs

The screenshot shows the 'Add permissions' step of the IAM role creation wizard. The user has selected the 'AmazonDynamoDBFullAccess' policy from a list of three matches. The policy is described as providing full access to Amazon DynamoDB.

~ Click **Next**.

3. Name the role:

Role name: **lambda-dynamodb-counter-role**

- Click “Create Role”.

The screenshot shows the 'Name, review, and create' step of the IAM role creation wizard. The user has entered 'lambda-dynamodb-counter-role' in the 'Role name' field and provided a descriptive text in the 'Description' field.

Identity and Access Management (IAM)

Summary

Creation date
November 20, 2025, 19:34 (UTC+05:30)

Last activity
2 hours ago

ARN
arn:aws:iam::820314685364:role/lambda-dynamodb-counter-role

Maximum session duration
1 hour

Permissions | Trust relationships | Tags | Last Accessed | Revoke sessions

Permissions policies (2)

You can attach up to 10 managed policies.

Policy name	Type	Attached entities
AmazonDynamoDBFullAccess	AWS managed	1
CloudWatchLogsFullAccess	AWS managed	1

AWS Lambda:

- AWS Lambda is a serverless compute service that runs your code automatically in response to events without provisioning or managing servers.
- It lets you execute code on-demand and scale automatically, with billing based only on compute time used.

Compute

AWS Lambda

lets you run code without thinking about servers.

You pay only for the compute time that you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration.

Get started

Author a Lambda function from scratch, or choose from one of many preconfigured examples.

Create a function

How it works

.NET | Java | **Node.js** | Python | Ruby | Custom runtime

```
1 * exports.handler = async (event) => {
2   console.log(event);
3   return 'Hello from Lambda!';
}
```

Run | Next: Lambda responds to events

Step:2 - Create Lambda Function:

1. Create function.

Choose: **Author from scratch**

Function name: **VisitorCounterFunction**

Runtime: **Node.js 22.x**

The screenshot shows the 'Create function' wizard in the AWS Lambda console. The 'Basic information' section is visible, containing fields for 'Function name' (set to 'VisitorCounterFunction'), 'Runtime' (set to 'Node.js 22.x'), and 'Architecture' (set to 'x86_64'). There are also sections for 'Permissions' and 'Additional configurations'.

2. Change “Execution Role” section: Select Use an existing role

- Choose the role you created: **lambda-dynamodb-counter-role**

- Click **Create Function**.

The screenshot shows the 'Permissions' section of the 'Create function' wizard. It includes a note about Lambda creating an execution role and a 'Change default execution role' section. Under 'Execution role', the 'Use an existing role' option is selected, and the role 'lambda-dynamodb-counter-role' is chosen from a dropdown menu. The 'Additional configurations' section at the bottom is also visible.

Step:3 - Add Lambda Code:

1. Go to → “Code” section
2. You will see a file called: **index.mjs (or index.js)**
3. Delete the default code.
4. Paste the below code.

```

```

import { DynamoDBClient, UpdateItemCommand } from "@aws-sdk/client-dynamodb";
const ddb = new DynamoDBClient({});

export const handler = async () => {
 // DynamoDB table and counter ID
 const TableName = "siteCounters";
 const CounterId = "visitors";
 // Atomic increment
 const params = {
 TableName,
 Key: { counterId: { S: CounterId } },
 UpdateExpression: "ADD #count :inc",
 ExpressionAttributeNames: { "#count": "count" },
 ExpressionAttributeValues: { ":inc": { N: "1" } },
 ReturnValues: "UPDATED_NEW",
 };
 const result = await ddb.send(new UpdateItemCommand(params));
 const totalVisitors = result.Attributes.count.N;
}

```

```

return {

 statusCode: 200,

 headers: { "Access-Control-Allow-Origin": "*" },

 body: JSON.stringify({ value: totalVisitors }),

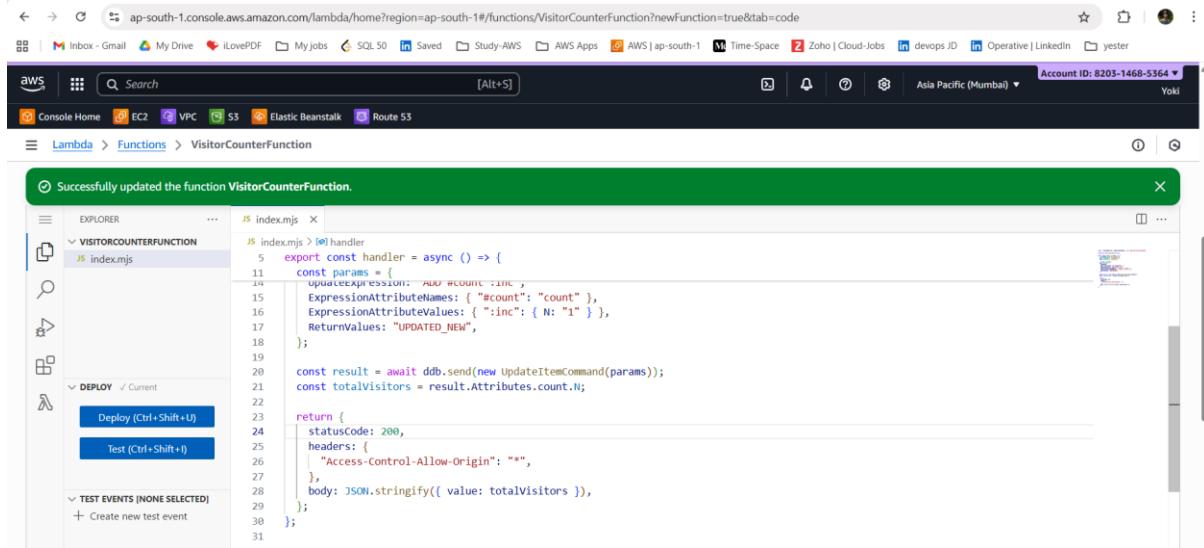
};

};

```

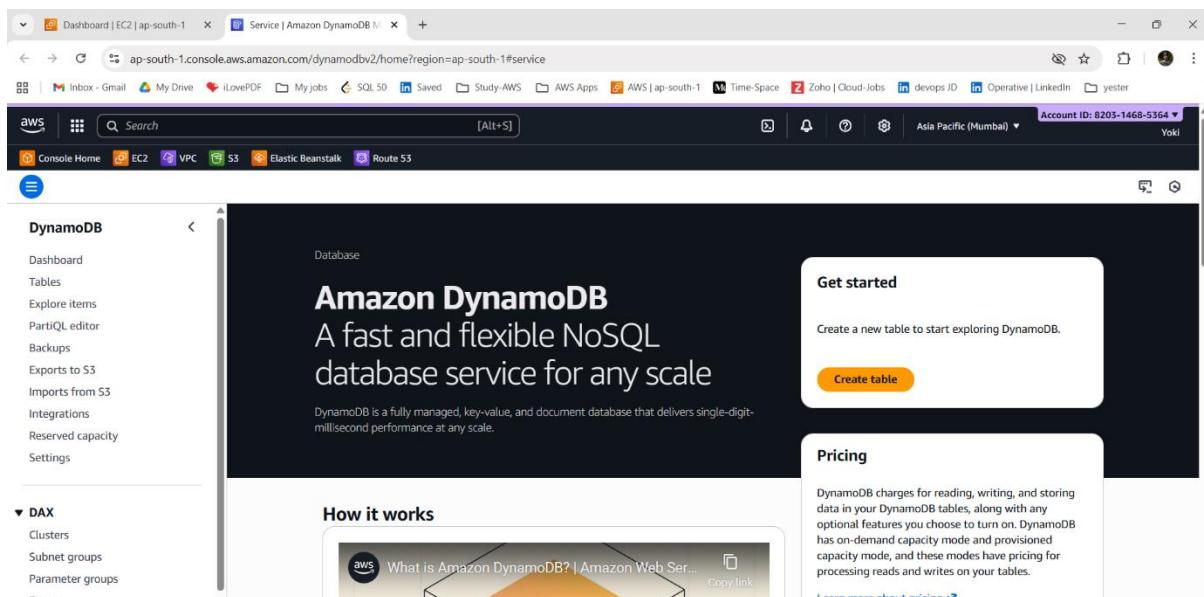
```

5. Click “Deploy”.



AMAZON DynamoDB:

- Amazon DynamoDB is a fully managed NoSQL database service from AWS that provides fast, predictable performance at any scale. It offers key-value and document data storage with built-in availability, security, and automatic scaling.



Step:4 - Create DynamoDB table:

1. Go to DynamoDB → Create table.

Table name: siteCounters

Partition key: counterId (String)

The screenshot shows the 'Create table' wizard in the AWS DynamoDB console. The 'Table details' step is active. The 'Table name' field contains 'siteCounters'. The 'Partition key' field is set to 'counterid' with a type of 'String'. The 'Sort key - optional' field is empty. The page includes standard AWS navigation and account information at the top.

2. Billing mode: Capacity mode = **On-demand**

The screenshot shows the 'Table settings' step in the 'Create table' wizard. It offers two options: 'Default settings' (selected) and 'Customize settings'. Under 'Default table settings', it lists various configuration options like Table class (DynamoDB Standard), Capacity mode (On-demand), and Maximum read/write capacity units (both set to '-'). The 'Customize settings' section is collapsed. The page includes standard AWS navigation and account information at the top.

3. click “Create table”.

The screenshot shows the AWS DynamoDB console with the 'Tables' page selected. On the left, there's a sidebar with options like 'Dashboard', 'Tables', 'PartQL editor', 'Exports to S3', 'Imports from S3', 'Integrations', 'Reserved capacity', and 'Settings'. The main area shows a table titled 'Tables (1)'. The table has one row for 'siteCounters'. The columns include 'Name' (siteCounters), 'Status' (Active), 'Partition key' (counterId \$), 'Sort key' (-), 'Indexes', 'Replication Regions', 'Deletion protection' (Off), 'Favorite', and 'Read capacity'. At the top right of the table view, there are buttons for 'Actions', 'Delete', and 'Create table'. The status bar at the bottom indicates the URL as <https://ap-south-1.console.aws.amazon.com/dynamodbv2/home?region=ap-south-1#partql-editor>.

4. Add the first item.

5. Go to: Items → Create item

counterId (Partition key) : visitors - string
count : 0 - number

6. Click “Create item”.

The screenshot shows the 'Create item' dialog for the 'siteCounters' table. At the top, it says 'Create item' and 'Form | JSON view'. Below that, a note says 'You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep.' There's a link to 'Learn more'. The 'Attributes' section contains two rows: 'counterId - Partition key' with value 'visitors' and type 'String', and 'count' with value '0' and type 'Number'. There are 'Add new attribute', 'Cancel', and 'Create item' buttons at the bottom. The status bar at the bottom indicates the URL as https://ap-south-1.console.aws.amazon.com/dynamodbv2/home?region=ap-south-1#edit-item?itemMode=1&route=ROUTE_ITEM_EXPLORER&table=siteCounters.

API GATEWAY:

- It is a fully managed service that lets you create, publish, secure, and monitor APIs at any scale.
- It acts as a front door for applications to access backend services like Lambda, EC2, or microservices.

The screenshot shows the AWS API Gateway homepage. At the top, there are navigation links for Console Home, EC2, VPC, S3, Elastic Beanstalk, and Route 53. The main heading is "API Gateway" with the subtext "Create and manage APIs at scale". Below this, a subtext states: "Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs." A "Get started" box contains the text: "Create a new API to begin exploring API Gateway. You can also import an external definition file into API Gateway." A "Create an API" button is present. On the left, a "How it works" diagram illustrates the flow from client devices (mobile, web) through API Gateway to various AWS services (Lambda, Kinesis, Step Functions, S3, CloudWatch, VPC, Data Center). On the right, a "Pricing" section explains the pay-as-you-go model for HTTP and REST APIs, mentioning no minimum fees or upfront commitments. It also notes that for WebSocket APIs, fees are based on the number of messages and connection duration. A "Learn more about pricing" link is provided. At the bottom, there are links for CloudShell, Feedback, Console Mobile App, Privacy, Terms, and Cookie preferences.

Step:5- Create API Gateway:

1. Go to API Gateway → Create API.

2. enter API details:

API name: **VisitorCounterAPI**

IP Address: IPv4

3. Integrations → Choose: **Lambda**

AWS Region: **ap-south-1**

Lambda Function: **arn:aws:lambda-function:VisitorCounter**

The screenshot shows the "Create HTTP API" wizard on the AWS API Gateway console. The browser address bar shows the URL: ap-south-1.console.aws.amazon.com/apigateway/main/create?region=ap-south-1. The page title is "VisitorCounterFunction | Function". The navigation bar includes links for EC2, VPC, S3, Elastic Beanstalk, and Route 53. The main content area is titled "API details" and shows the following configuration:

- API name:** VisitorCounterAPI
- IP address type:** IPv4 (selected)
- Integrations (1) Info:** A Lambda integration is selected, with the Lambda function set to "arn:aws:lambda:ap-south-1:820314685364:function:VisitorCounter".

On the left, a sidebar lists steps: Step 2 - optional (Configure routes, Define stages), Step 3 - optional (Define stages), Step 4, and Review and create. At the bottom, there are links for CloudShell, Feedback, Console Mobile App, Privacy, Terms, and Cookie preferences.

5. Configure routes:

Method:	GET
Resource path:	/count
Integration Target:	VisitorCounterFunction.

6. Define stages = default

7. Review and create.

The screenshot shows the AWS API Gateway console. In the top navigation bar, it says "API Gateway > APIs > VisitorCounterAPI (m8k4ijx2z7)". A green success message box at the top right says "Successfully created API VisitorCounterAPI (m8k4ijx2z7)." On the left, there's a sidebar with sections like "APIs", "Custom domain names", "Domain name access associations", "VPC links", and a main section for the "VisitorCounterAPI" with sub-sections "Develop" and "Deploy". The "Default endpoint" is listed as "Enabled" with the URL "https://m8k4ijx2z7.execute-api.ap-south-1.amazonaws.com".

Step:6- Copy Endpoint:

1. In your API → go to Stages → Select default.

2. Copy the Invoke URL.

e.g., <https://abcdedf.execute-api.ap-south-1.amazonaws.com> .

3. Open your browser/Postman and test:

4. And add the route path “/count”.

<https://m8k4ijx2z7.execute-api.ap-south-1.amazonaws.com/count>

→ You should see something like:

{"value":"2"}

The screenshot shows a browser window with multiple tabs open. The active tab is for the URL "https://m8k4ijx2z7.execute-api.ap-south-1.amazonaws.com/count". The page content displays the JSON response {"value": "2"}.

OUTCOME:

Here's an outcome description of the project *before hosting the website on S3*, based on the steps listed:

- ~ The project successfully set up a scalable backend system by creating an IAM role, building a Lambda function, and adding the necessary code to process visitor counts.
- ~ A database table was created to store visit data, and APIs were configured to allow secure access to the counter.
- ~ The API endpoint was generated and ready to be integrated into a frontend or S3-hosted static website.

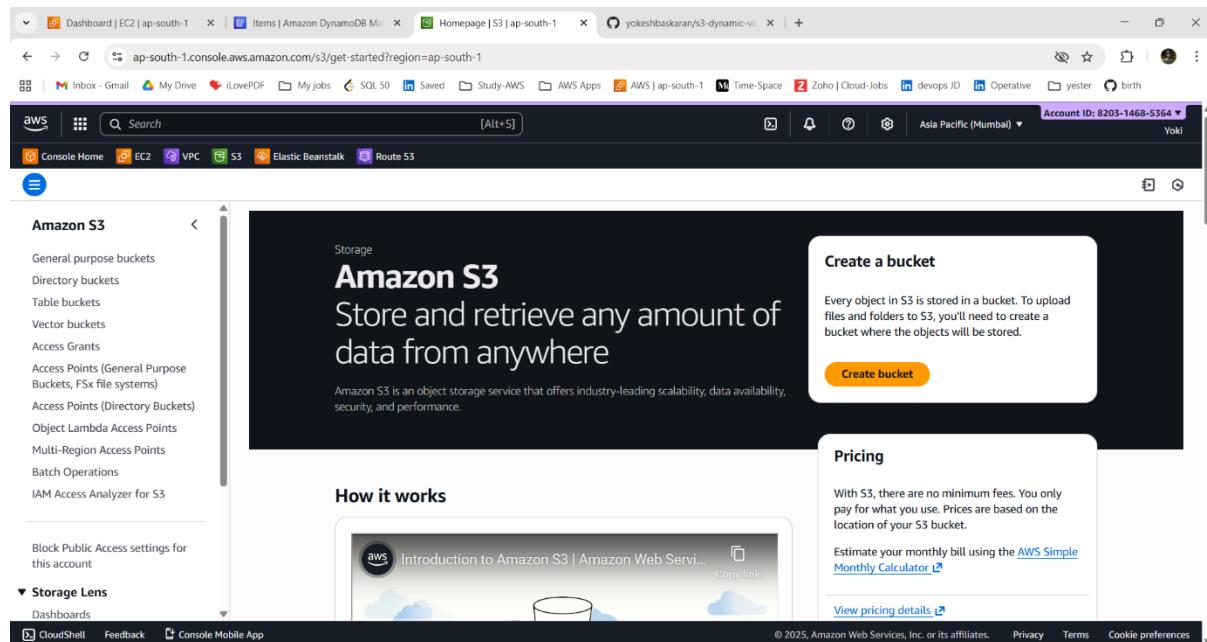
STATIC WEBSITE HOSTING ON AMAZON S3:

~ project-name: **s3-dynamic-visitor-counter**

- This document describes how to host a static website using Amazon S3's built-in static hosting feature.
- This approach is suitable for simple static sites that do not require HTTPS, backend logic, authentication, or private hosting.

S3 STATIC HOSTING:

- S3 static hosting in AWS is a feature that lets you host static websites (HTML, CSS, JavaScript, images) directly from an Amazon S3 bucket without needing a web server.
- It serves your site via HTTP/HTTPS with high availability and low cost.



Here, we will see how to host a static website using Amazon S3's built-in static hosting feature. This approach is suitable for simple static sites that do not require HTTPS, backend logic, authentication, or private hosting.

~ Go → AWS Management Console and Open the S3 service.

1. Create an S3 Bucket:

- i) Choose “Create bucket”.
- ii) Enter the bucket name (must be globally unique).

Bucket-name: **s3-dynamic-visitor-counter**

The screenshot shows the 'Create bucket' wizard. In the 'General configuration' section, the 'Bucket name' is set to 's3-dynamic-visitor-counter'. The 'Bucket type' dropdown is set to 'General purpose'. A note indicates that general purpose buckets are recommended for most use cases. The 'AWS Region' is set to 'Asia Pacific (Mumbai) ap-south-1'. The 'Copy settings from existing bucket - optional' section shows a single bucket selected. The 'Choose bucket' button is visible. The URL in the browser is ap-south-1.console.aws.amazon.com/s3/buckets?region=ap-south-1.

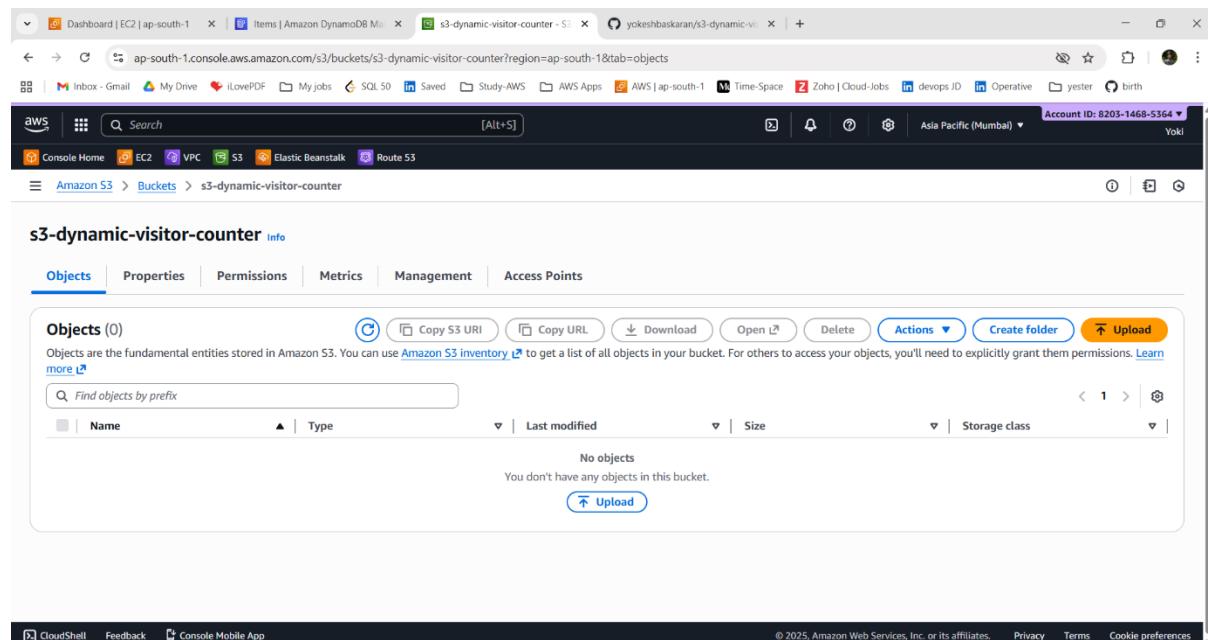
- iii) Select an AWS Region - **ap-south-1**

iv) Create the bucket.

The screenshot shows the 'Amazon S3 > Buckets' page. A green success message at the top states 'Successfully created bucket "s3-dynamic-visitor-counter"'. Below it, there are two tabs: 'General purpose buckets' (selected) and 'Directory buckets'. The 'General purpose buckets' table lists three buckets: 's3-dynamic-visitor-counter', 's3-static-bday-countdown', and 'user-counter'. Each entry includes the bucket name, AWS Region (Asia Pacific (Mumbai) ap-south-1), and creation date. To the right of the table are three callout boxes: 'Account snapshot' (updated daily), 'External access summary - new' (updated daily), and a note about Storage Lens. The bottom of the page includes links for CloudShell, Feedback, and Console Mobile App, along with copyright information and links for Privacy, Terms, and Cookie preferences.

2. Upload Your Website Files:

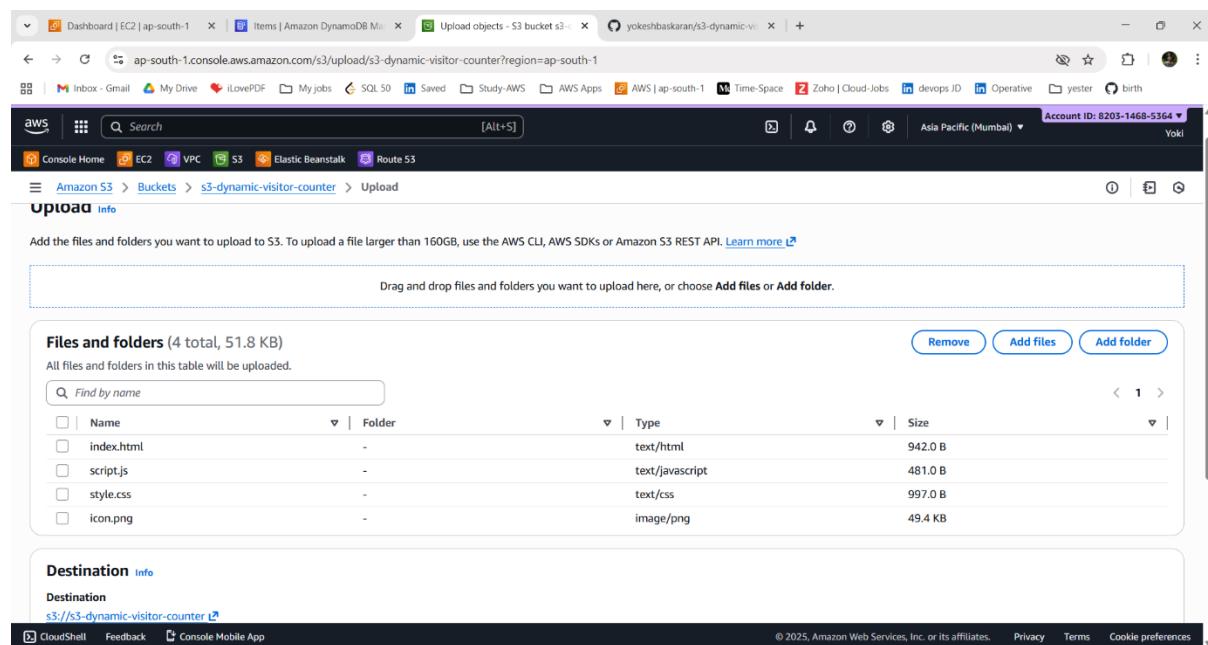
- i) Open the s3-dynamic-visitor-counter bucket.



The screenshot shows the AWS S3 console interface. The top navigation bar includes links for EC2, Amazon DynamoDB, and the current S3 bucket. The main title is 's3-dynamic-visitor-counter'. Below it, there are tabs for 'Objects', 'Properties', 'Permissions', 'Metrics', 'Management', and 'Access Points'. The 'Objects' tab is active, displaying a message: 'No objects. You don't have any objects in this bucket.' At the bottom of the objects list is a blue 'Upload' button. The bottom of the screen features a dark footer bar with links for CloudShell, Feedback, and Console Mobile App, along with copyright information and privacy terms.

- ii) Choose **Upload** to upload the files.

- iii) Add all files related to your website (e.g., index.html, error.html, CSS, JS files).



The screenshot shows the 'Upload objects' page for the 's3-dynamic-visitor-counter' bucket. The 'Upload' tab is active. A large text input field is labeled 'Drag and drop files and folders you want to upload here, or choose Add files or Add folder.' Below this is a table titled 'Files and folders (4 total, 51.8 KB)'. The table lists four files: 'index.html' (text/html, 942.0 B), 'script.js' (text/javascript, 481.0 B), 'style.css' (text/css, 997.0 B), and 'icon.png' (image/png, 49.4 KB). To the right of the table are three buttons: 'Remove', 'Add files', and 'Add folder'. At the bottom left is a 'Destination' section with a dropdown menu set to 's3://s3-dynamic-visitor-counter'. The bottom of the screen features a dark footer bar with links for CloudShell, Feedback, and Console Mobile App, along with copyright information and privacy terms.

iv) Choose **Upload** again to complete the process.

Name	Type	Last modified	Size	Storage class
icon.png	png	November 21, 2025, 12:41:25 (UTC+05:30)	49.4 KB	Standard
index.html	html	November 21, 2025, 12:41:24 (UTC+05:30)	942.0 B	Standard
script.js	js	November 21, 2025, 12:41:24 (UTC+05:30)	481.0 B	Standard
style.css	css	November 21, 2025, 12:41:25 (UTC+05:30)	997.0 B	Standard

3. Enable Static Website Hosting:

- i) Go to the “Properties” tab of the bucket.
- ii) Scroll to **Static website hosting**.
- iii) Choose **Edit** and then Select **Enable**.
- iv) Choose Host a static website.
- v) Set your Index document (e.g., **index.html**).
- vi) click **Save changes**.

Static website hosting

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

Enable

Host a static website

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.

`index.html`

Error document - optional

This is returned when an error occurs.

`error.html`

vii) Copy the Website endpoint shown after saving — this is your public website URL.

URL: <http://s3-dynamic-visitor-counter.s3-website.ap-south-1.amazonaws.com>

The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with 'General purpose buckets' (Directory buckets, Table buckets, Vector buckets, Access Grants, Access Points (General Purpose Buckets, FSx file systems), Access Points (Directory Buckets), Object Lambda Access Points, Multi-Region Access Points, Batch Operations, IAM Access Analyzer for S3) and 'Storage Lens' (Dashboards). The main area shows the 's3-dynamic-visitor-counter' bucket under 'Buckets'. A green success message at the top says 'Successfully edited static website hosting.' Below it, the 'Static website hosting' section is visible, with a note about using AWS Amplify Hosting for static website hosting. It shows 'S3 static website hosting' is enabled and 'Hosting type' is 'Bucket hosting'. Under 'Bucket website endpoint', it displays the URL <http://s3-dynamic-visitor-counter.s3-website.ap-south-1.amazonaws.com>. The bottom right of the page has links for 'Privacy', 'Terms', and 'Cookie preferences'.

~ If we get 403 Forbidden or access denied. We need to check the bucket settings.

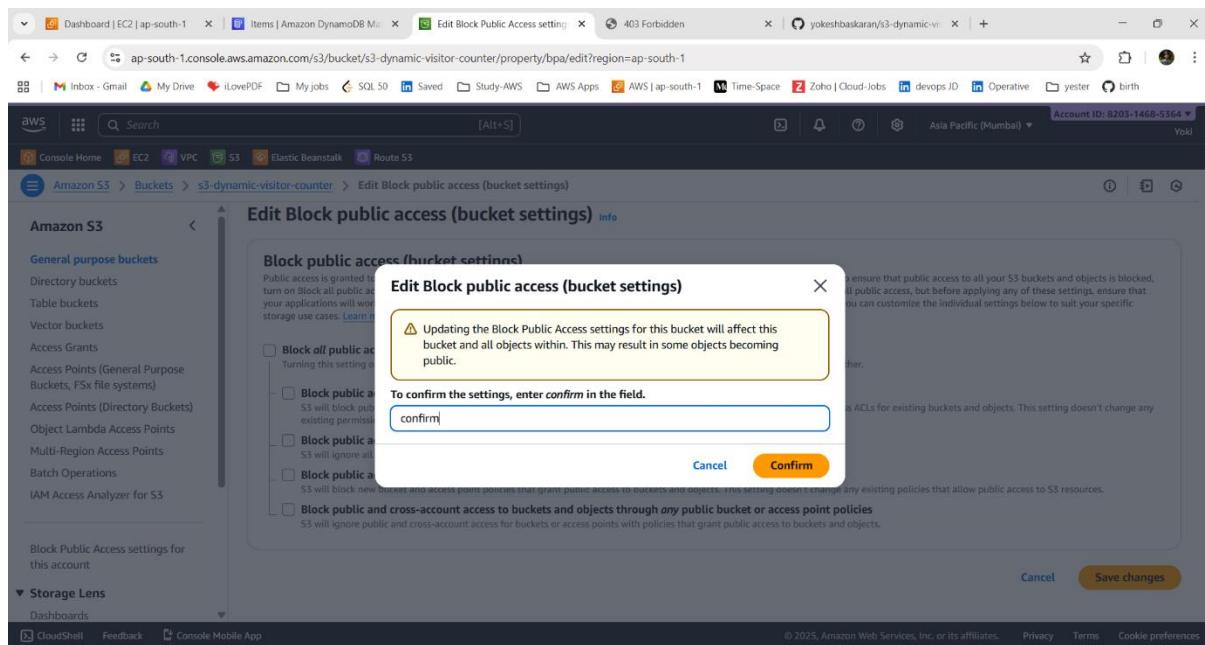
The screenshot shows a browser window with a 403 Forbidden error page. The URL in the address bar is 'Not secure s3-dynamic-visitor-counter.s3-website.ap-south-1.amazonaws.com'. The error message '403 Forbidden' is displayed prominently. Below it, there's a list of error details: Code: AccessDenied, Message: Access Denied, RequestId: N5RQTM515W7SGTFZ, and HostId: 185LtbXlvjvT4vtzVahBQT0UEM4CW5fX/Rgit/2+aF5ior7Td8Y34ZCjI0WykhTy7buBA9Fih0Ip9VDRgtqoPDEbP5CGglo.

~ we need to **Turn off “Block Public Access”** (bucket settings).

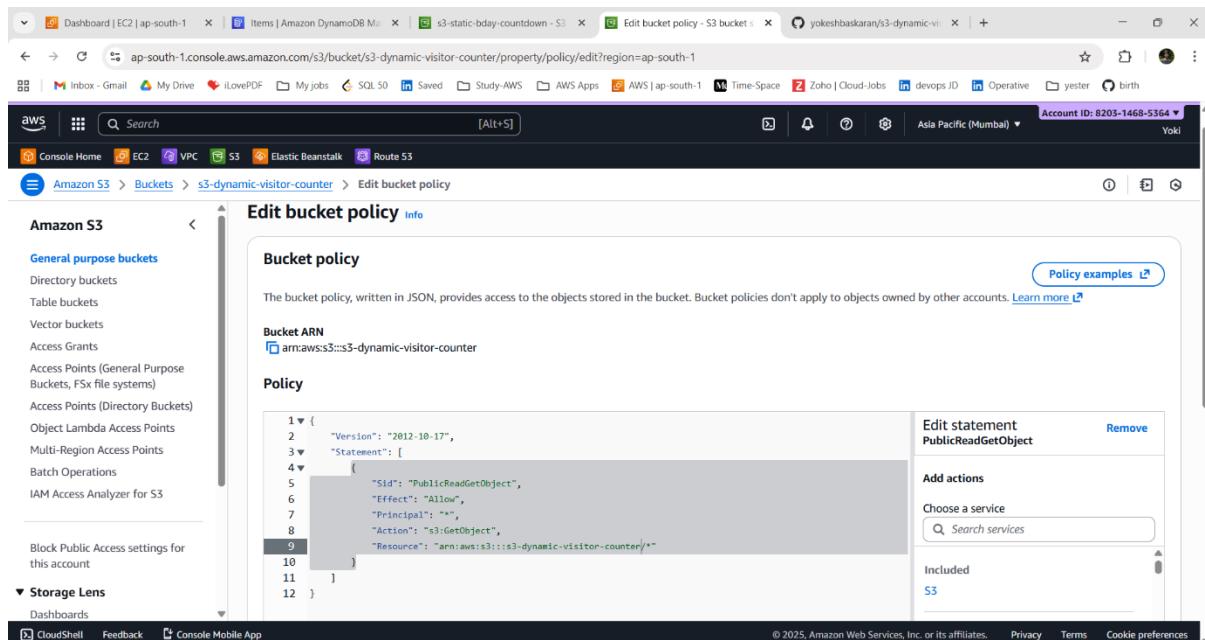
~ Also, give the **Bucket policy**.

4. Allow Public Access to Website Files and add Bucket Policy:

- i) Go to the “Permissions” tab of the bucket.
- ii) Scroll to “Block Public Access” (bucket settings).
- iii) Choose **Edit** and **uncheck** the “Block all Public Access”.
- iv) type “confirm” and click Save Changes.



- v) In same page of “Permissions” tab of the bucket. Scroll to **Bucket policy**.
- vi) Choose **Edit** and add a policy that allows public read of all objects in the bucket.



BUCKET POLICY:

```

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "PublicReadGetObject",

"Effect": "Allow",

"Principal": "\*",

"Action": "s3:GetObject",

"Resource": "arn:aws:s3:::s3-dynamic-visitor-counter/\*"

}

]

}

```

vii) Save the policy.

viii) Verify that Public Access indicators show that the bucket allows public reads.

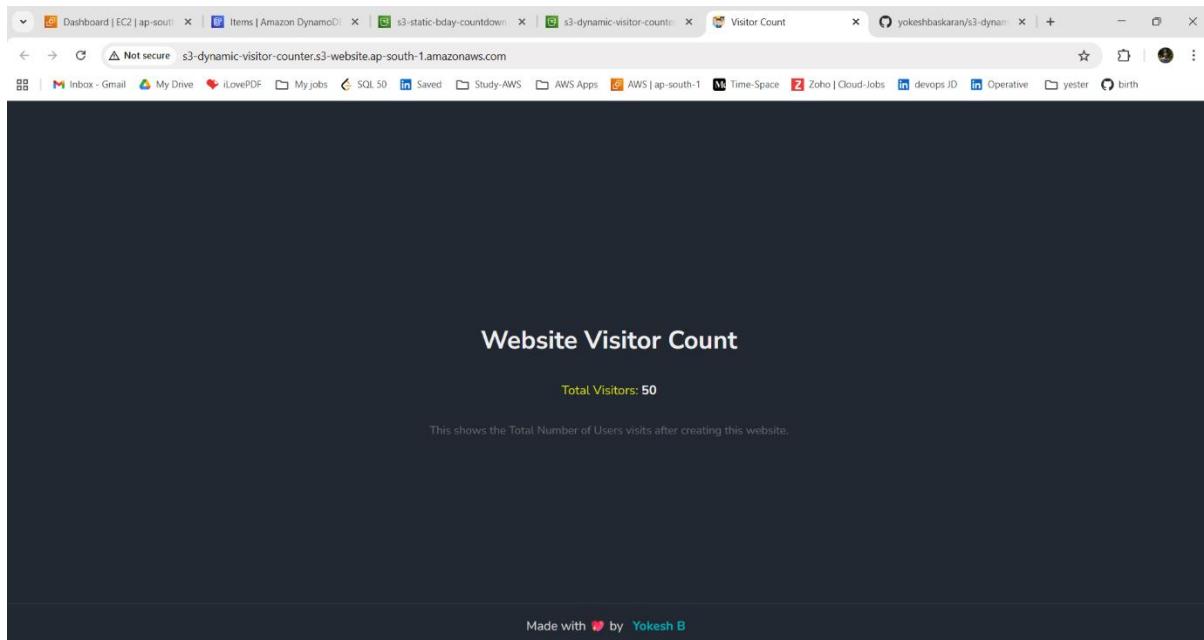
The screenshot shows the AWS S3 Bucket Policy editor. At the top, there's a green success message: "Successfully edited bucket policy." Below it, the "Bucket policy" section displays the JSON policy code. The policy grants public read access to all objects in the bucket. The JSON code is as follows:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::s3-dynamic-visitor-counter/*"
    }
  ]
}
```

On the right side of the editor, there are "Edit" and "Delete" buttons, and a "Copy" button. The left sidebar shows the "Amazon S3" navigation menu with options like "General purpose buckets", "Storage Lens", and "CloudShell". The main navigation bar at the top includes links for EC2, VPC, S3, and Route 53.

5. Verify Website Accessibility:

- i) Open the S3 Website endpoint URL in your browser.
- ii) Confirm that the site loads and that resources (CSS, JS files) work properly.
- iii) If something doesn't load, check:
 - Files are uploaded at the correct paths
 - Permissions allow public read
 - File names match exactly (case-sensitive)

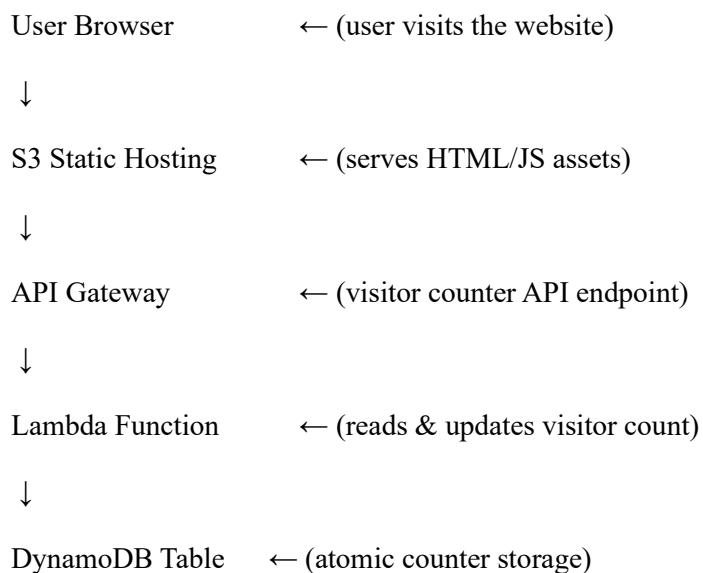


6. Maintenance & Updates:

- i) To update your site, upload new versions of your files to the S3 bucket.
 - ii) Overwrite existing objects or delete and re-upload.
 - iii) Refresh the website to view changes (sometimes a hard refresh is needed: Ctrl+Shift+R).
- ~ updated code version.

A screenshot of the Amazon S3 'Upload' interface. The top navigation bar shows 'Amazon S3 > Buckets > s3-dynamic-visitor-counter > Upload'. The main area has a 'Upload info' section with instructions to add files and a 'Drag and drop files and folders you want to upload here, or choose Add files or Add folder.' button. Below this is a 'Files and folders' table showing four items: index.html, script.js, style.css, and icon.png. The table includes columns for Name, Folder, Type, and Size. At the bottom, there's a 'Destination' section with the URL 's3://s3-dynamic-visitor-counter' and links for CloudShell, Feedback, and Console Mobile App. The footer includes copyright information and links for Privacy, Terms, and Cookie preferences.

VISUAL ARCHITECTURE DIAGRAM:



CONCLUSION

The project successfully implemented a **scalable serverless website visitor counter** using AWS Lambda, API Gateway, DynamoDB, and S3. The solution eliminates server management, scales automatically with traffic, and stores visitor data reliably.

By combining these AWS services, the system delivers a lightweight, cost-efficient, and highly available way to track website visits in real time, demonstrating the effectiveness of serverless architecture for modern web applications.