Fully functioning Web application using serverless technologies

AWS Lambda

API Gateway

Amazon S3

Amazon DynamoDB

Amazon Simple Email Service (SES)

IAM Roles

1. Project Overview

Company Name: Belly Brew Co. Limited
Industry: Manufacturing and distribution of gut-friendly beverages and ready meals
Market: Retail supermarkets, grocery stores, restaurants, and coffee shops across the United Kingdom
Mission Statement: Innovating gut-friendly meals and beverages that fuse nutrition and convenience, fostering well-being with every choice.

---

2. Problem Statement

Belly Brew Co. Limited seeks to expand customer awareness and brand loyalty while capturing additional market share in the ready meals and soft drinks industry. The company aims to engage customers by providing an enticing offer through a website, encouraging users to sign up for a newsletter in exchange for a free gut-friendly recipe eBook. The ultimate goal is to create a secure, scalable, and manageable cloud-based solution to gather customer details and facilitate email marketing campaigns.

---

3. Business Requirements

Primary Goal:

- **Customer Engagement and Market Expansion**: Capture the names and email addresses of both existing and new customers interested in Belly Brew Co.'s product range.
- **Brand Awareness**: Build brand loyalty and customer awareness across the ready meals and beverages industry.

Key Requirements:

1. **Customer Information Capture**: A web-based solution to collect names and email addresses from customers.
2. **Newsletter Subscription Form**: The website should have a subscription form that entices users to sign up for the newsletter in exchange for a free gut-friendly recipe eBook.
3. **Email Campaigns**: The gathered email addresses will be used for future marketing campaigns.
4. **E-book Delivery**: Once the user signs up for the newsletter, the system must deliver a free recipe eBook via email.

---

4. Key Stakeholder Insights

- **Ellie Brew (CEO)**:
  The company's marketing department requires a cloud solution to capture customer details (name and email). The solution should work in collaboration with Belly Brew Co.'s internal development team.
- **Anisha Belly (Head of Marketing)**:
  The marketing team wants a website with a subscription form for visitors to sign up for the newsletter and receive a free eBook with over 100 recipes. The captured email addresses will be used for future marketing campaigns.
- **George Gut (CTO)**:
  The company seeks a cloud-based, serverless solution that is secure, scalable, and requires no administrative or management overhead. The solution should ensure that only genuine signups

receive the eBook, avoiding fraudulent or automated submissions. The company also aims to avoid the overhead of managing and maintaining servers.

---

5. Technical Requirements

Solution Characteristics:

- **Serverless Architecture**: A serverless solution will avoid the administrative overhead of managing servers and ensure automatic scaling, security, and resilience.
- **Data Storage**: Customer details (names and email addresses) should be securely stored in a database for marketing purposes.
- **Email Automation**: A system must be in place to automatically email the eBook to subscribers immediately after sign-up.
- **Security**: The system must validate that only genuine users who sign up receive the eBook.

---

6. Solution Design Overview

High-Level Workflow:

1. **User Interaction**: Customers, such as Gemma, visit the Belly Brew Co. website and subscribe to the newsletter by submitting their name and email.
2. **Form Submission**: Upon submission, the data is processed by a serverless compute service that stores customer details in a database (e.g., DynamoDB) for marketing purposes.
3. **Email Delivery**: An automated email is sent to the user with a secure link to download the recipe eBook.
4. **E-Book Access**: The customer receives the eBook and can download it from the link provided in the email, ensuring a positive user experience while securing the file from unauthorized access.

---

7. Conclusion

The proposed solution for Belly Brew Co. Limited addresses their need to capture customer data and deliver a value proposition through an automated, serverless cloud architecture. By implementing this solution, Belly Brew Co. can increase customer engagement, market share, and brand loyalty while keeping operational costs low and security high.

1. Reducing Infrastructure Management Overhead

- **Serverless** technologies like AWS Lambda are utilized to reduce the need for managing infrastructure.
- **Amazon S3** is used to host the static website and the eBook itself, avoiding the need for EC2 instances or managing servers.
- **Amazon DynamoDB** provides a no-SQL database solution, avoiding the need for database administration (such as with RDS).
- **Amazon API Gateway** facilitates communication between frontend and backend services, decoupling components and ensuring scalable, manageable APIs.

2. Ensuring Secure eBook Delivery

- The eBook is stored in an **S3 bucket** and accessed via a **pre-signed URL** generated dynamically.
- This URL is time-limited, preventing widespread sharing while still providing user-friendly access.
- **Amazon Simple Email Service (SES)** handles the email functionality, securely sending the pre-signed URL link to users.

3. Detailed Process Overview

- **User Interaction:** The customer submits a form via the static website.
- **Lambda Functions**: Triggered by API Gateway, Lambda performs business logic, stores user details in DynamoDB, and generates the pre-signed URL.

- **Email Delivery:** SES sends the pre-signed URL, ensuring a smooth flow from signup to delivery with time-restricted access to the eBook.

4. Additional Considerations

- **Security**: IAM roles are configured to ensure each service (Lambda, API Gateway, S3, etc.) has the necessary permissions, keeping the application secure.
- **Cost Efficiency**: Using serverless technologies like Lambda and S3 hosting allows for cost savings, especially when services are idle.

This architecture ensures that you meet the key requirement of reducing infrastructure overhead while still delivering a secure, scalable solution for the eBook distribution. It also provides a clear separation of concerns between the frontend and backend, leveraging AWS's fully managed services to handle all key tasks without manual intervention.

To summarize the steps for setting up this serverless web application:

1. **Create an S3 bucket** for storing your recipe book. Make sure the bucket is private and upload your book.pdf file in the correct folder structure.
2. **Set up a DynamoDB table** for storing the user profiles (name and email). Use email as the partition key and leave the default capacity settings.
3. **Configure Amazon SES (Simple Email Service)** in a sandbox environment, and verify an email identity that will be used to send emails.
4. **Create an IAM Policy and Role** for AWS Lambda. The policy should allow actions on the S3 bucket, DynamoDB, and SES services, and should be attached to the Lambda role.
5. **Create a Lambda function** using Python 3.9. Paste the provided Lambda code and configure the necessary environment variables such as the email address, S3 bucket name, DynamoDB table name, and region.
6. **Set up API Gateway** with an HTTP API, integrating it with the Lambda function. Define the /submit path for the POST method, which will trigger the Lambda function when the form is submitted.

7. HTML & API Integration:
You modified the HTML front-end, specifically line 4 and 5, to include the API Gateway invoke URL (with /submit appended). This URL triggers the backend flow when the form on the static website is submitted, sending the user's data to API Gateway.

8. S3 Setup:
You created an S3 bucket to host your static website. After setting the correct region, you ensured that public access was allowed by unchecking the block public access option and acknowledging that the bucket would be publicly accessible. Then, you uploaded all the static website files (HTML, CSS, JS, images, etc.).

9. Permissions & Policy:
Although public access was enabled, you still needed to apply an appropriate bucket policy. Using the AWS Policy Generator, you created an S3 bucket policy that allows public access to GET the objects (files) in your bucket. You included both the bucket ARN and its contents in the policy to make everything accessible.

10. Static Website Hosting:
You enabled static website hosting in the bucket properties, specifying the **index** and **error** pages. The index.html serves as the main page, while the error page is displayed if something goes wrong (e.g., file not found).

11. Testing:
After setting up the website and integrating the API, you tested the form submission process by filling in the form (name and email) and clicking the submit button. This action triggered the API Gateway, which invoked the Lambda function.

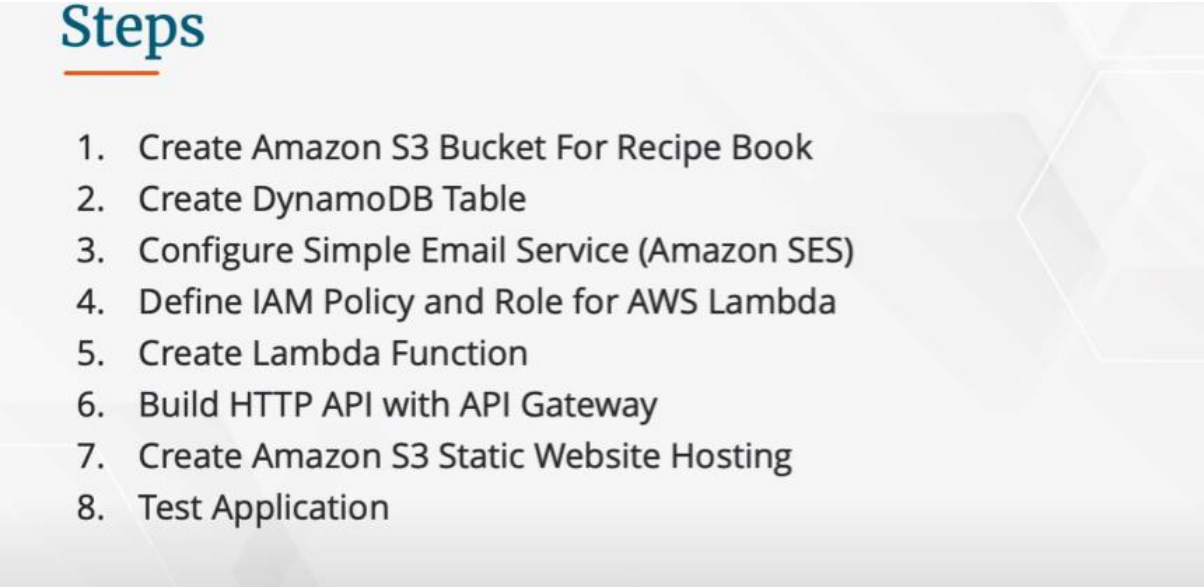
12. Lambda Execution:
The Lambda function processed the form submission by:

- Inserting the user’s name and email into a DynamoDB table.
- Generating a pre-signed URL for the ebook stored in S3.
- Using Amazon SES (Simple Email Service) to send an email to the user with the download link for the ebook.

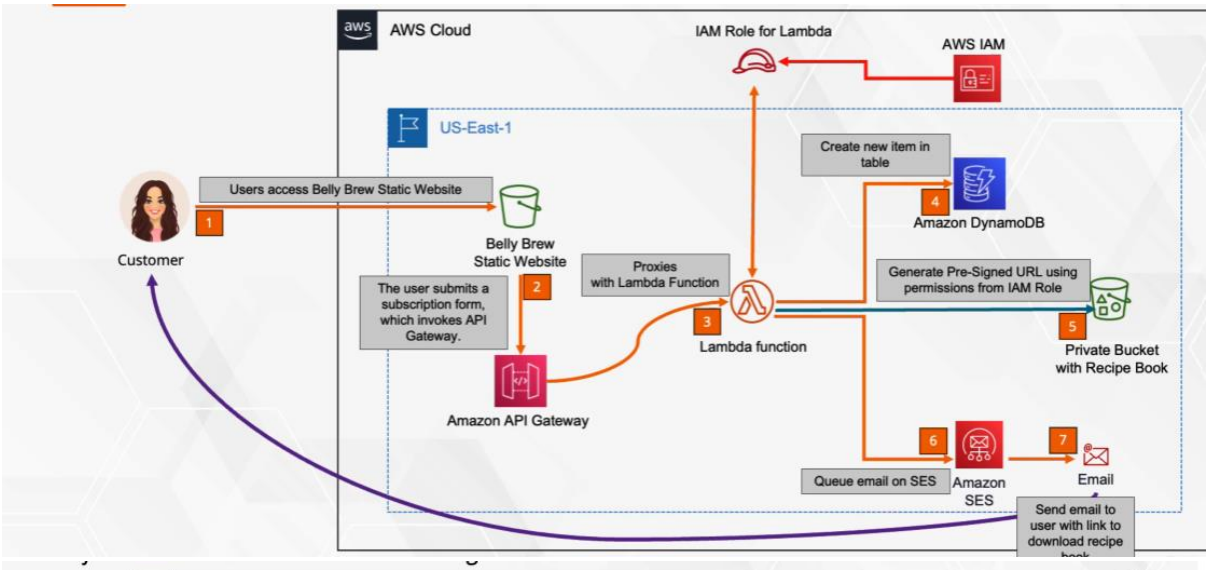
13. Monitoring & Verification:
You checked the **Lambda** function's monitoring tab to confirm that it was

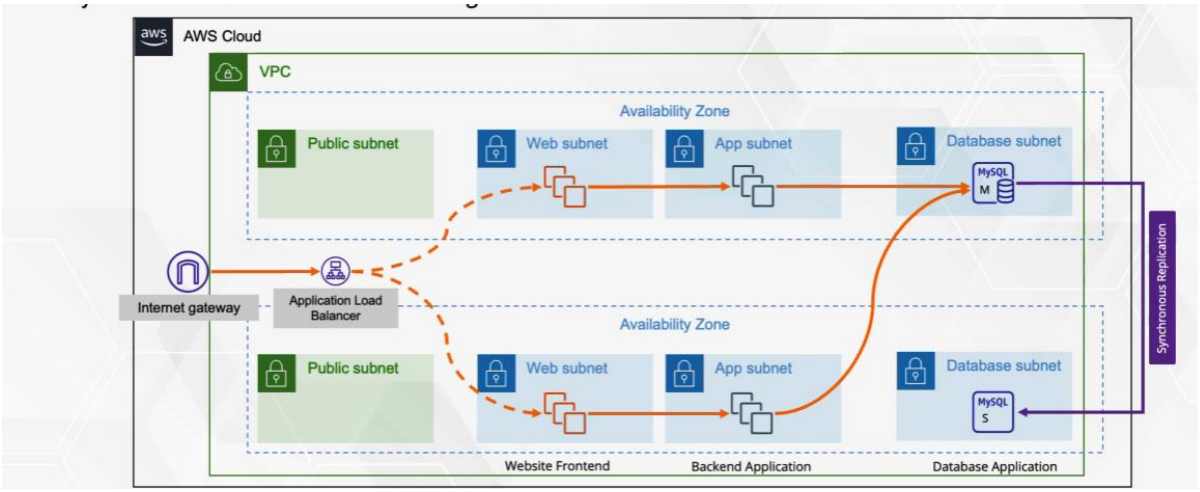
invoked when the form was submitted. You also verified in **DynamoDB** that the user's details were stored successfully, ensuring that the system captured the user's information as expected.

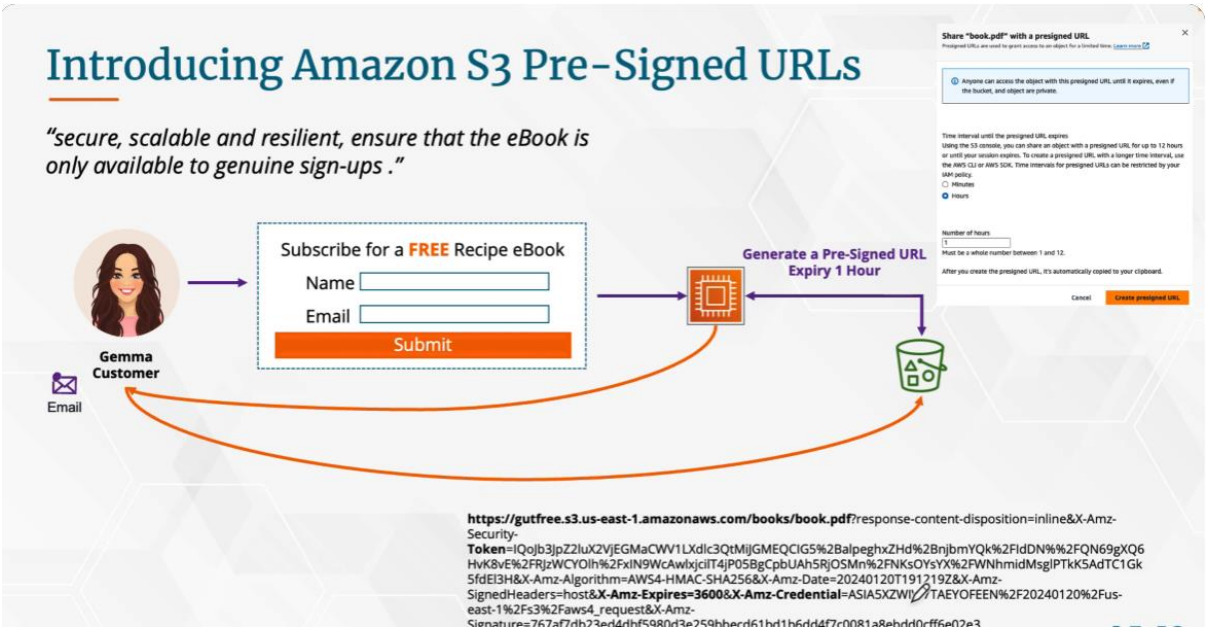
14. Final Output:
The user received an email containing a link to download the ebook, completing the process. This demonstrated how the entire serverless architecture—from front-end submission to backend processing—was working seamlessly.



ARCHITECTURE

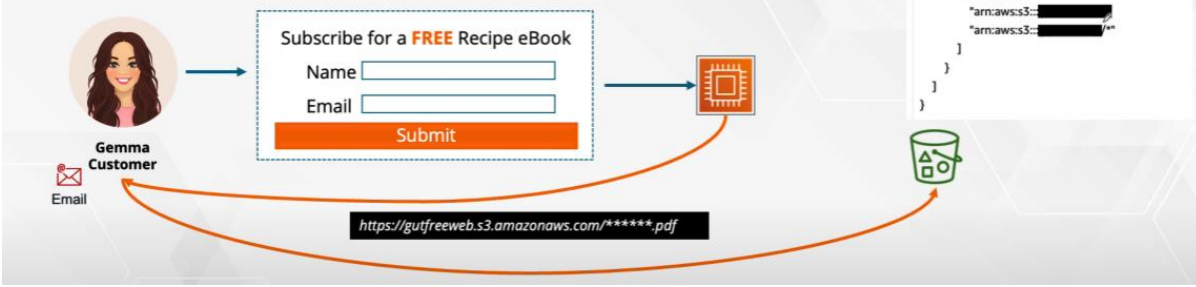




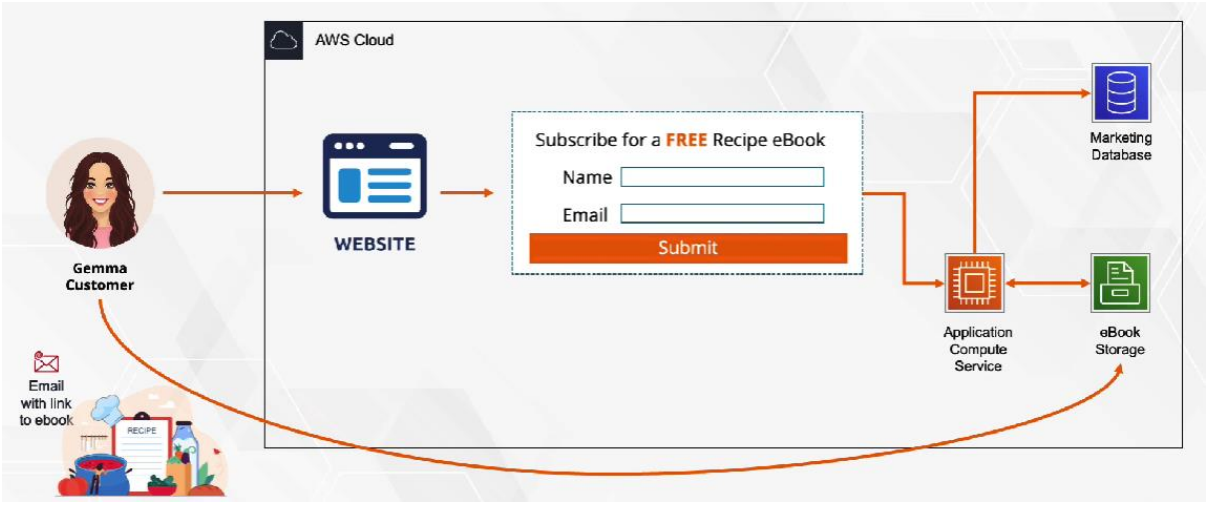


Fulfilling the business requirements

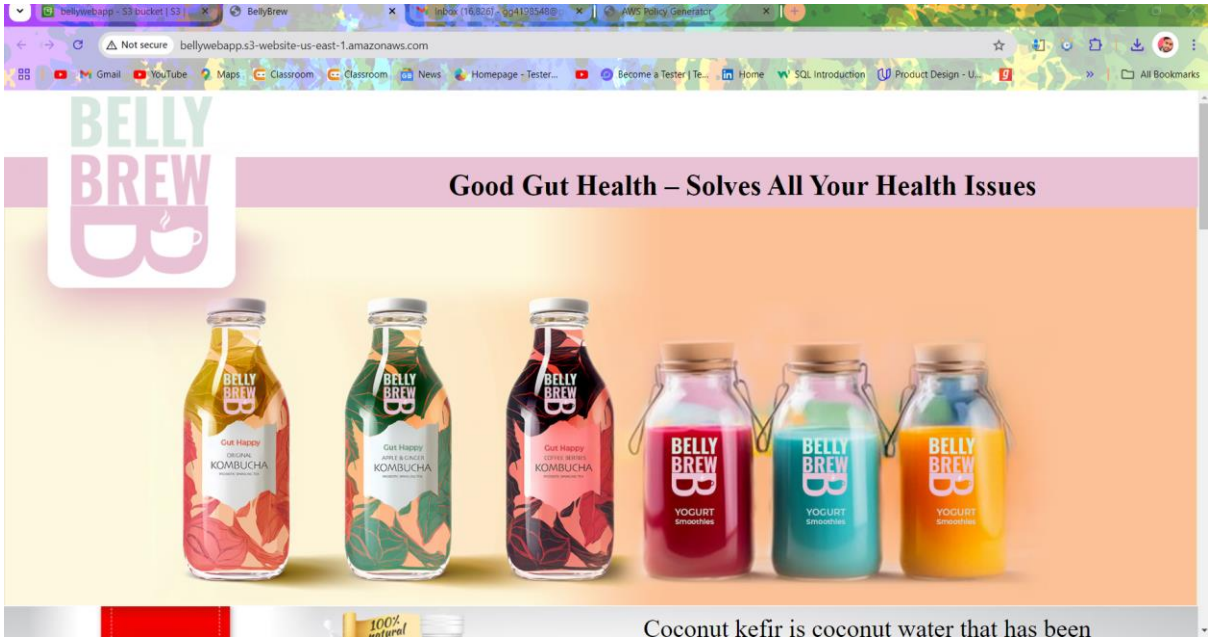
"secure, scalable and resilient, ensure that the eBook is only available to genuine sign-ups ."

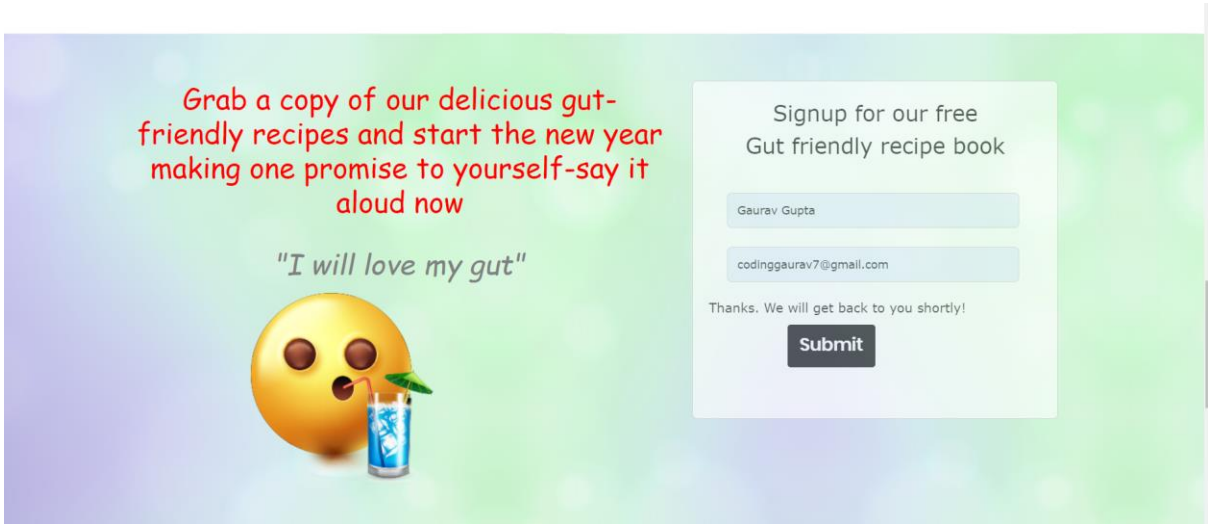


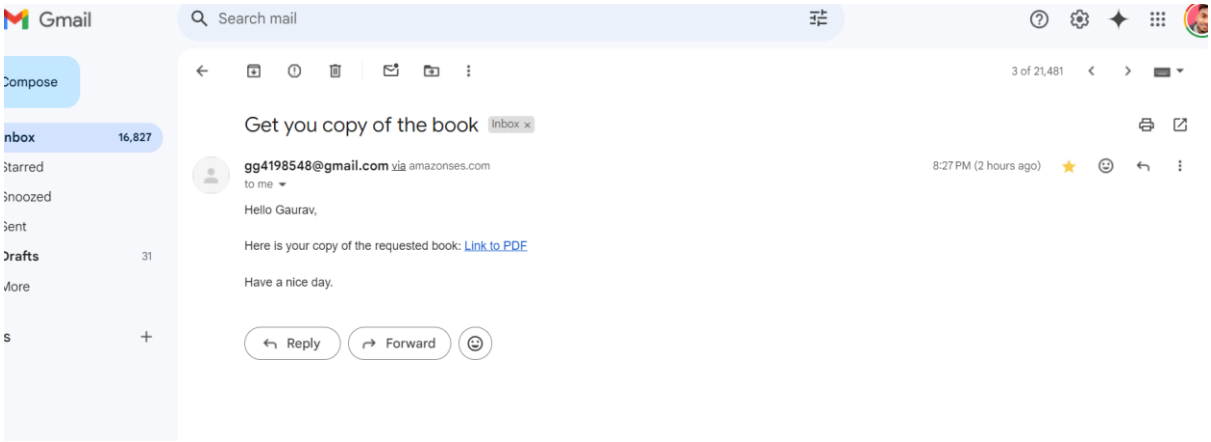

```
{
  "Version": "2012-10-17",
  "Id": "Policy1705424029637",
  "Statement": [
    {
      "Sid": "Stmt1705424028548",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": [
        "arn:aws:s3:::[redacted]",
        "arn:aws:s3:::[redacted]/*"
      ]
    }
  ]
}
```

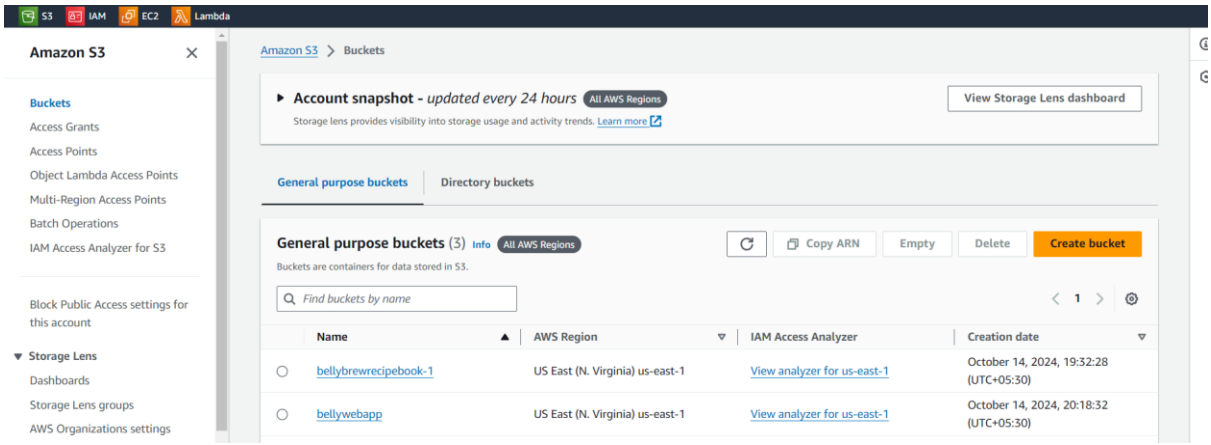
Deployed Website

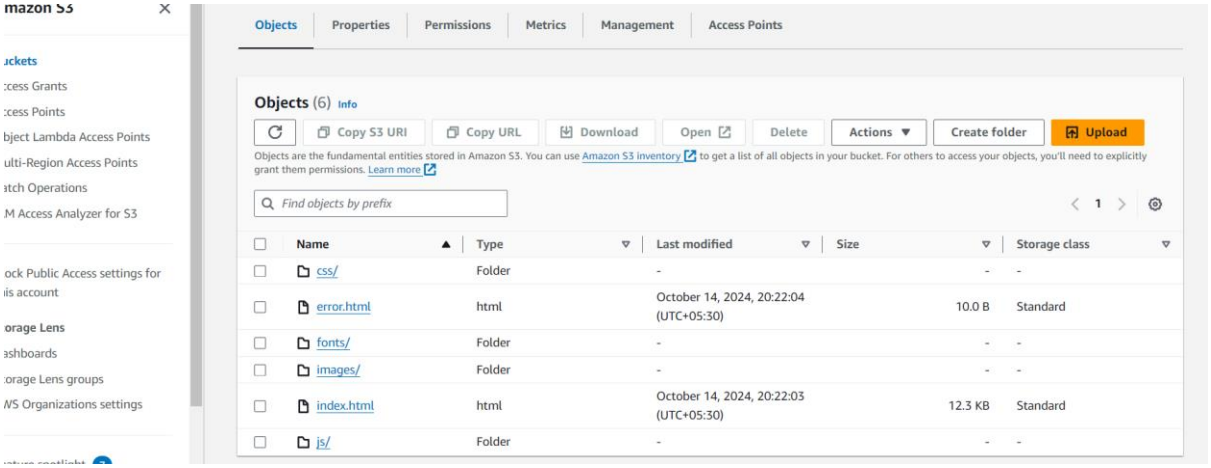


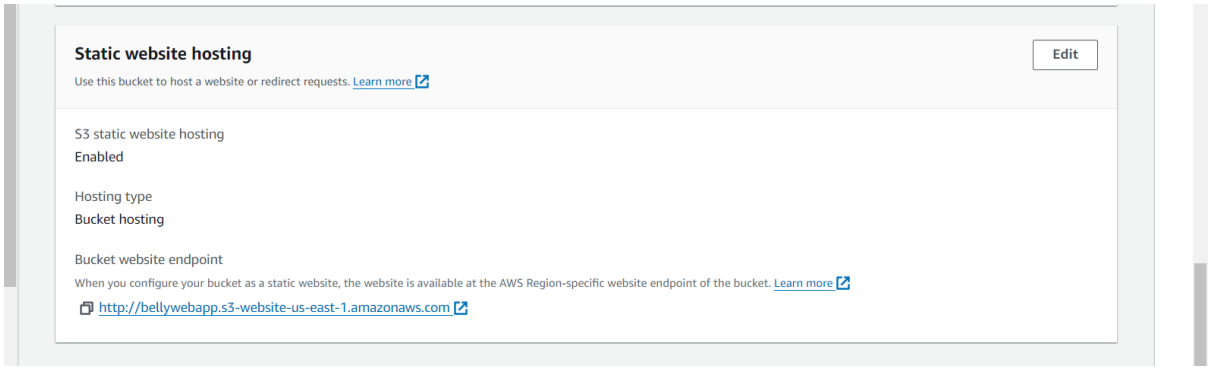


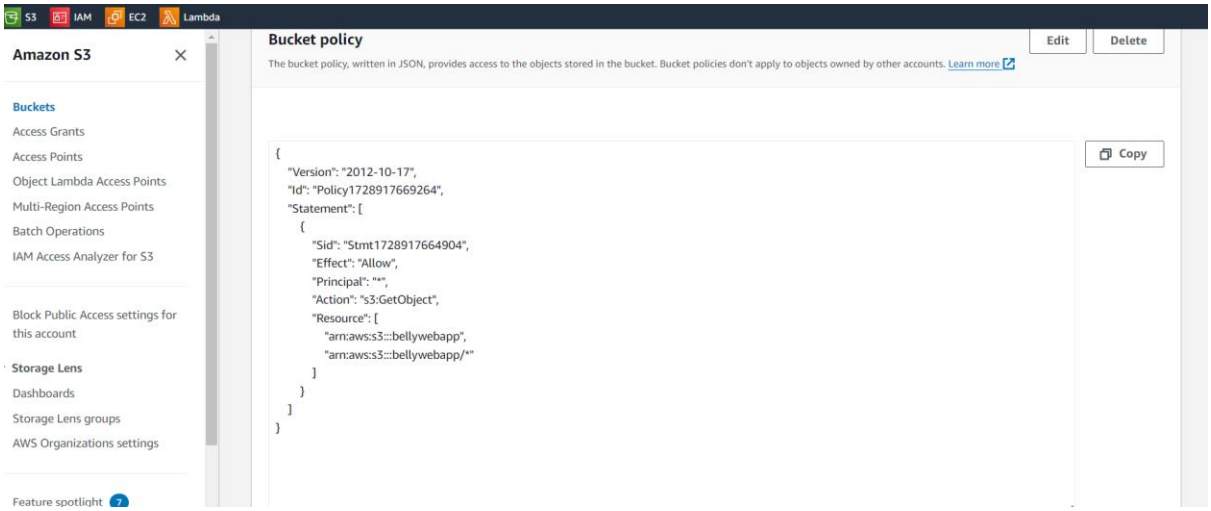


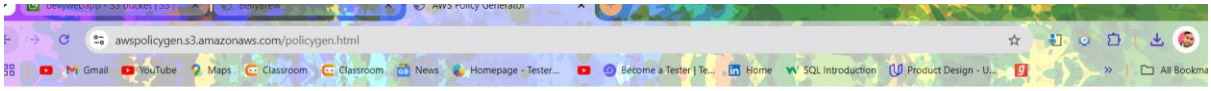
S3


S3 IAM EC2 Lambda
Amazon S3
Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings
Amazon S3 > Buckets
Account snapshot - updated every 24 hours
All AWS Regions
Storage lens provides visibility into storage usage and activity trends. Learn more
View Storage Lens dashboard
General purpose buckets
Directory buckets
General purpose buckets (3) Info
All AWS Regions
Buckets are containers for data stored in S3.
Copy ARN
Empty
Delete
Create bucket
Find buckets by name
Name
AWS Region
IAM Access Analyzer
Creation date
bellybrewrecipebook-1
US East (N. Virginia) us-east-1
View analyzer for us-east-1
October 14, 2024, 19:32:28 (UTC+05:30)
bellywebapp
US East (N. Virginia) us-east-1
View analyzer for us-east-1
October 14, 2024, 20:18:32 (UTC+05:30)


Objects
Properties
Permissions
Metrics
Management
Access Points
Objects (6) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
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
Find objects by prefix
Name
Type
Last modified
Size
Storage class
css/
Folder
error.html
html
October 14, 2024, 20:22:04 (UTC+05:30)
10.0 B
Standard
fonts/
Folder
images/
Folder
index.html
html
October 14, 2024, 20:22:03 (UTC+05:30)
12.3 KB
Standard
js/
Folder


Static website hosting
Edit
Use this bucket to host a website or redirect requests. Learn more
S3 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://bellywebapp.s3-website-us-east-1.amazonaws.com





amazon web services

AWS Policy Generator

The AWS Policy Generator is a tool that enables you to create policies that control access to Amazon Web Services (AWS) products and resources. For more information about creating policies, see key concepts in Using AWS Identity and Access Management. Here are sample policies.

Step 1: Select Policy Type

A Policy is a container for permissions. The different types of policies you can create are an IAM Policy, an S3 Bucket Policy, an SNS Topic Policy, a VPC Endpoint Policy, and an SQS Queue Policy.

Select Type of Policy S3 Bucket Policy

Step 2: Add Statement(s)

A statement is the formal description of a single permission. See a description of elements that you can use in statements.

Effect Allow Deny

Principal

Use a comma to separate multiple values.

AWS Service

Amazon S3

All Services ('*')

Use multiple statements to add permissions for more than one service.

Actions -- Select Actions -- All Actions ('*')

Amazon Resource Name (ARN)

Actions -- Select Actions -- All Actions ('*')

Amazon Resource Name (ARN)

ARN should follow the following format: arn:aws:s3:::${BucketName}/${KeyName}.
Use a comma to separate multiple values.

Add Conditions (Optional)

Add Statement

You added the following statements. Click the button below to Generate a policy.

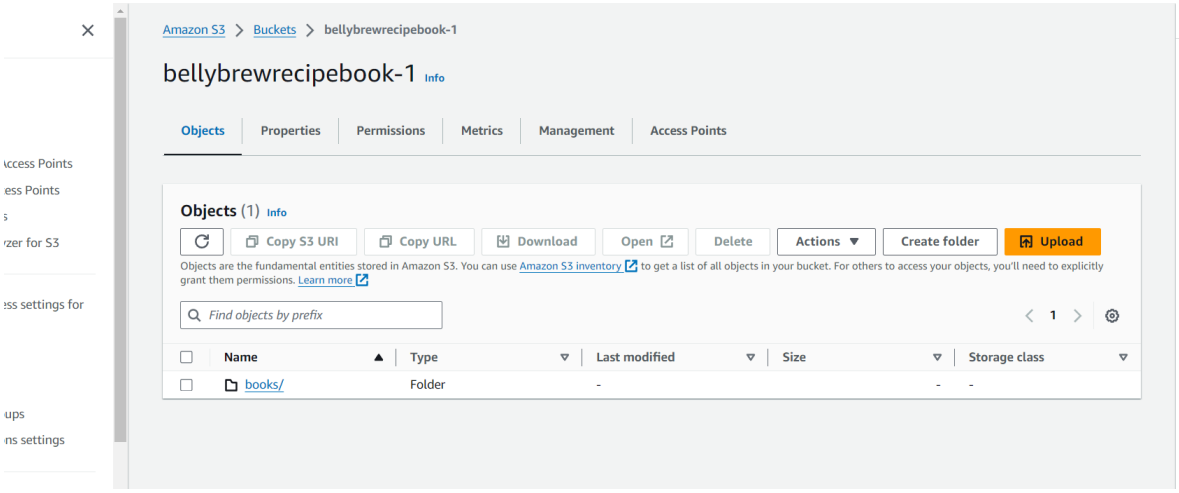
Principal(s)	Effect	Action	Resource	Conditions
• *	Allow	• s3:GetObject	• arn:aws:s3:::bellywebapp • arn:aws:s3:::bellywebapp/*	*None*

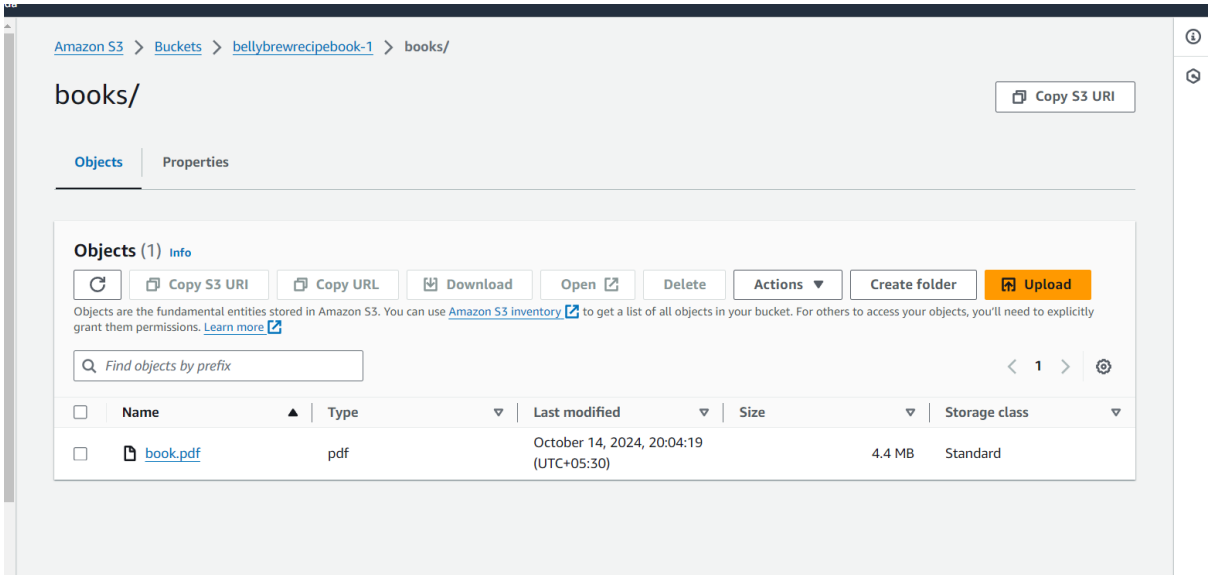
Step 3: Generate Policy

A *policy* is a document (written in the Access Policy Language) that acts as a container for one or more statements.

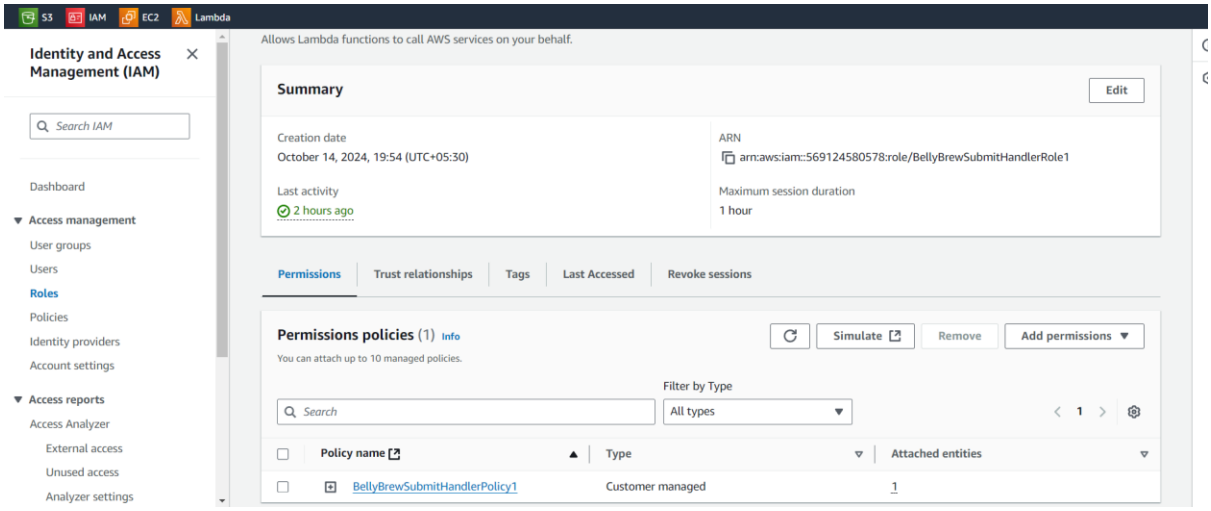
Generate Policy Start Over

Books

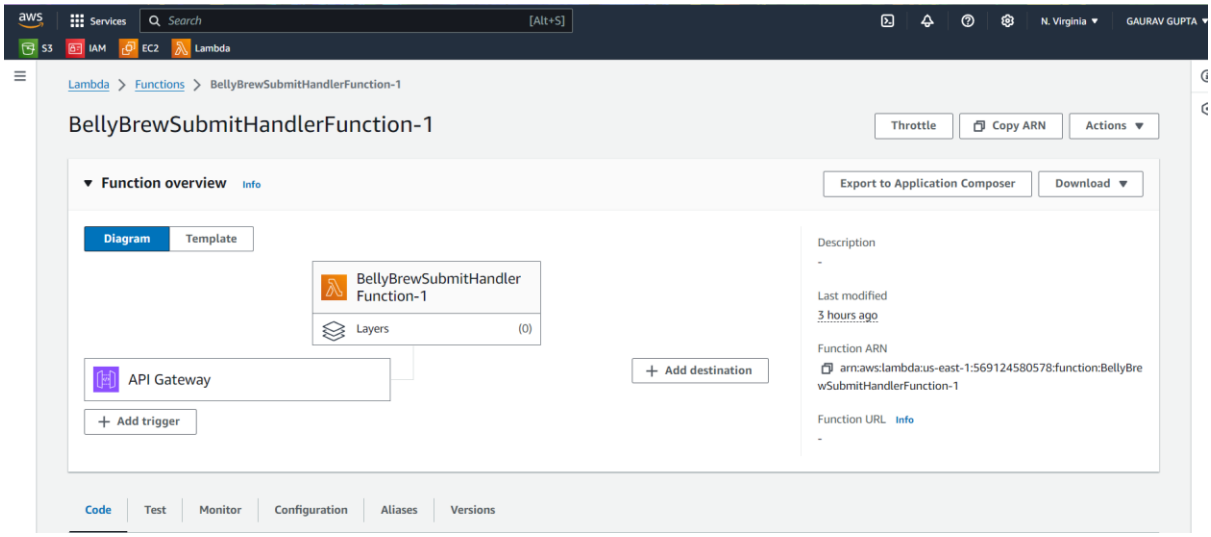

Amazon S3 > Buckets > bellybrewrecipebook-1
bellybrewrecipebook-1 Info
Objects
Properties
Permissions
Metrics
Management
Access Points
Objects (1) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 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
Find objects by prefix
Name
Type
Last modified
Size
Storage class
books/
Folder

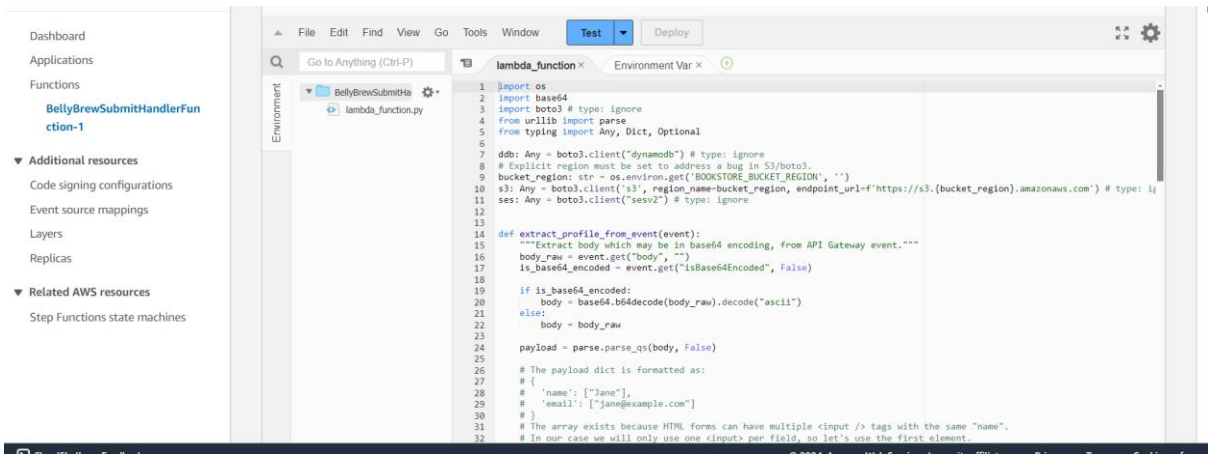

Amazon S3 > Buckets > bellybrewrecipebook-1 > books/
books/
Copy S3 URI
Objects
Properties
Objects (1) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
Name
Type
Last modified
Size
Storage class
book.pdf
pdf
October 14, 2024, 20:04:19 (UTC+05:30)
4.4 MB
Standard

IAM

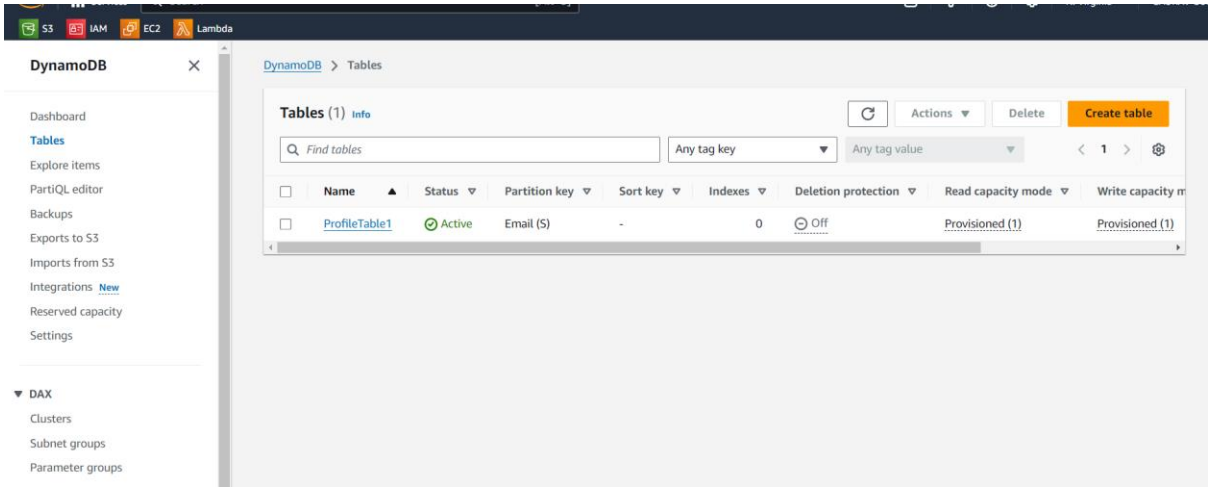


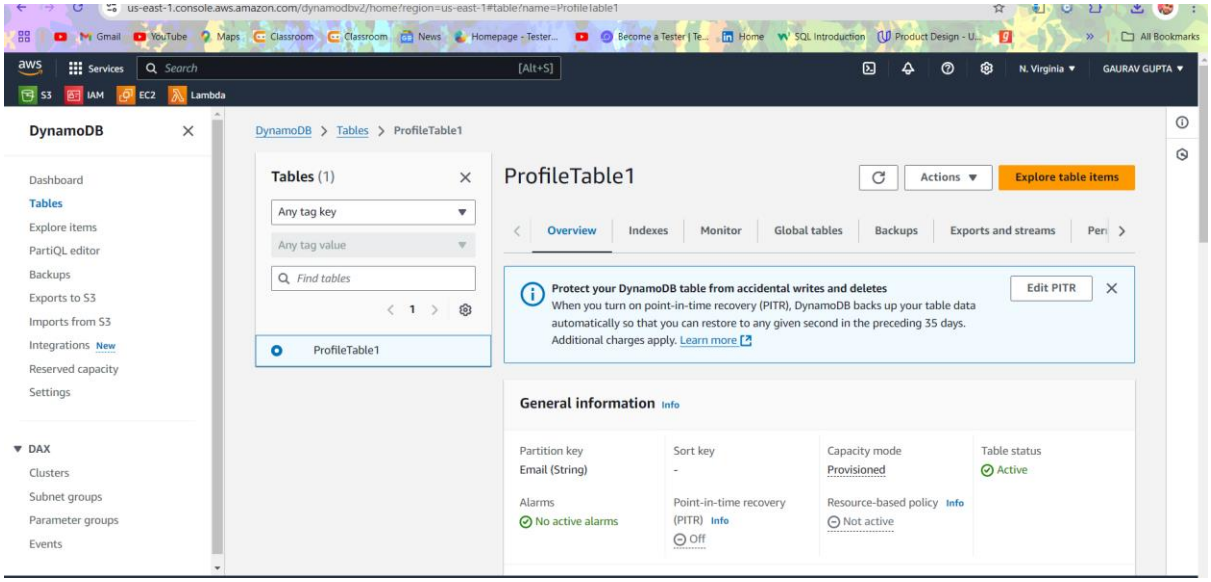
Lambda



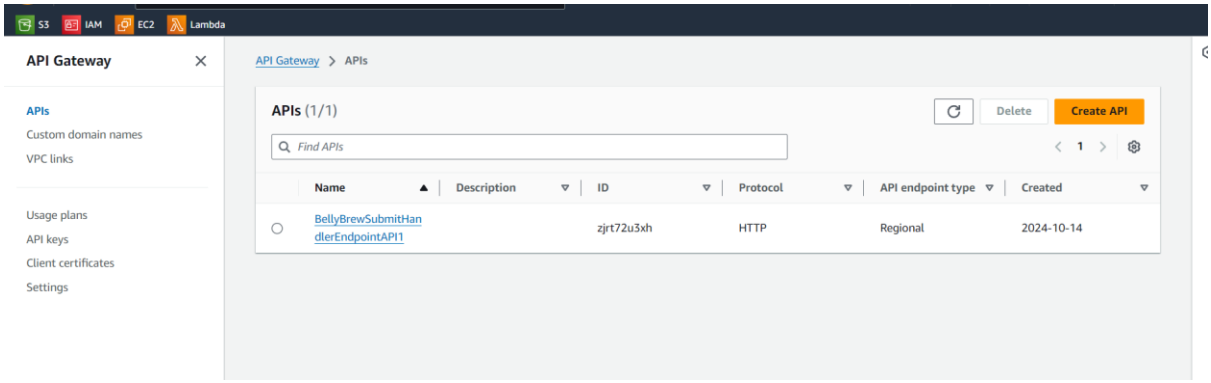


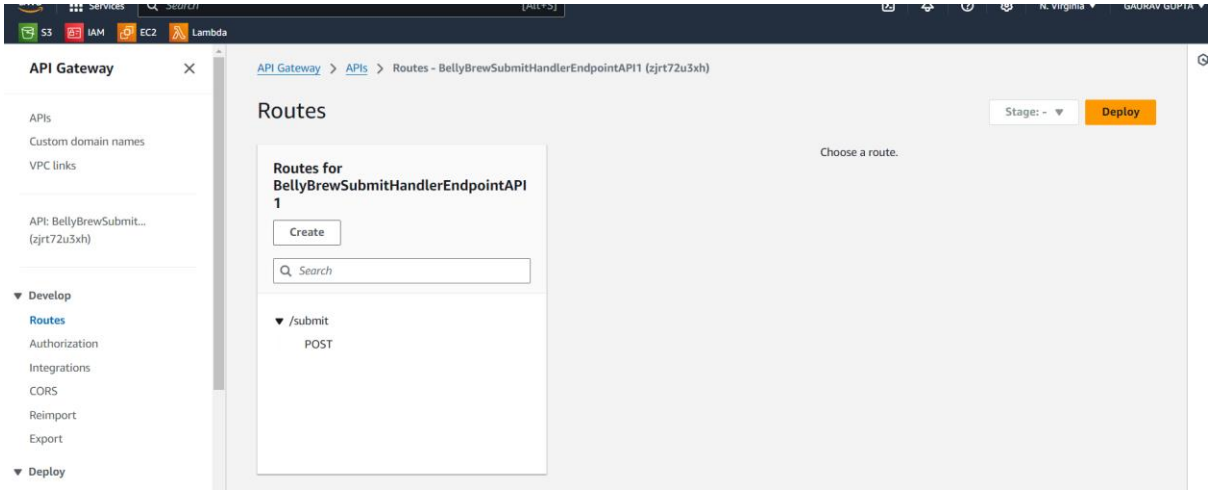
DynamoDB


S3
IAM
EC2
Lambda
DynamoDB
DynamoDB > Tables
Tables (1) Info
Actions
Delete
Create table
Find tables
Any tag key
Any tag value
Name
Status
Partition key
Sort key
Indexes
Deletion protection
Read capacity mode
ProfileTable1
Active
Email (S)
0
Off
Provisioned (1)
Dashboard
Tables
Explore items
PartiQL editor
Backups
Exports to S3
Imports from S3
Integrations New
Reserved capacity
Settings
DAX
Clusters
Subnet groups
Parameter groups

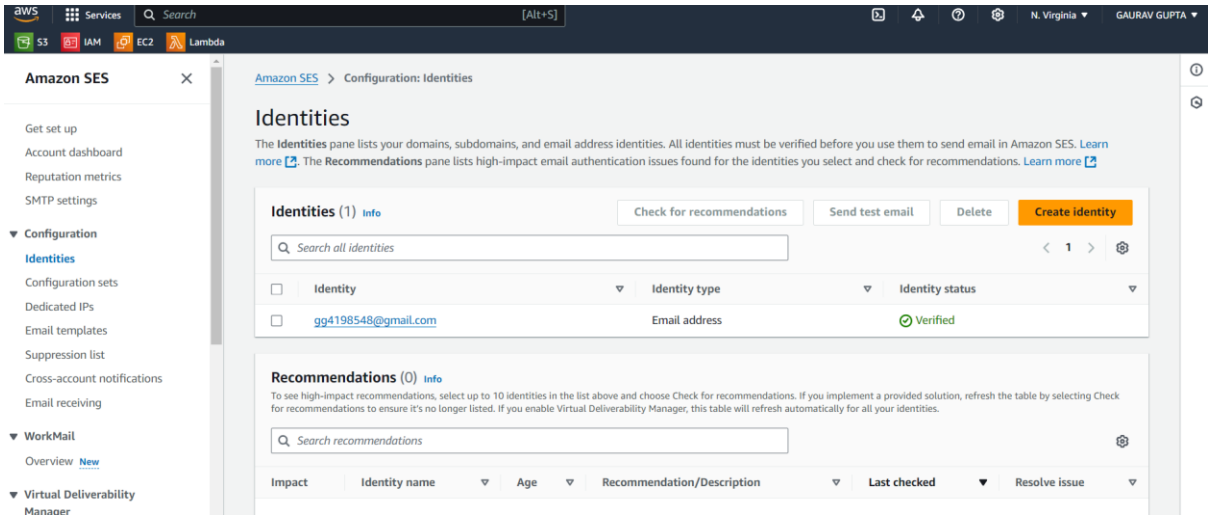

us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#table?name=ProfileTable1
aws
Services
Search
[Alt+S]
N. Virginia
GAURAV GUPTA
DynamoDB
DynamoDB > Tables > ProfileTable1
Tables (1)
ProfileTable1
Actions
Explore table items
Overview
Indexes
Monitor
Global tables
Backups
Exports and streams
Protect your DynamoDB table from accidental writes and deletes
When you turn on point-in-time recovery (PITR), DynamoDB backs up your table data automatically so that you can restore to any given second in the preceding 35 days.
Additional charges apply. Learn more
Edit PITR
General information Info
Partition key
Email (String)
Sort key
-
Capacity mode
Provisioned
Table status
Active
Alarms
No active alarms
Point-in-time recovery (PITR) Info
Off
Resource-based policy Info
Not active
Events

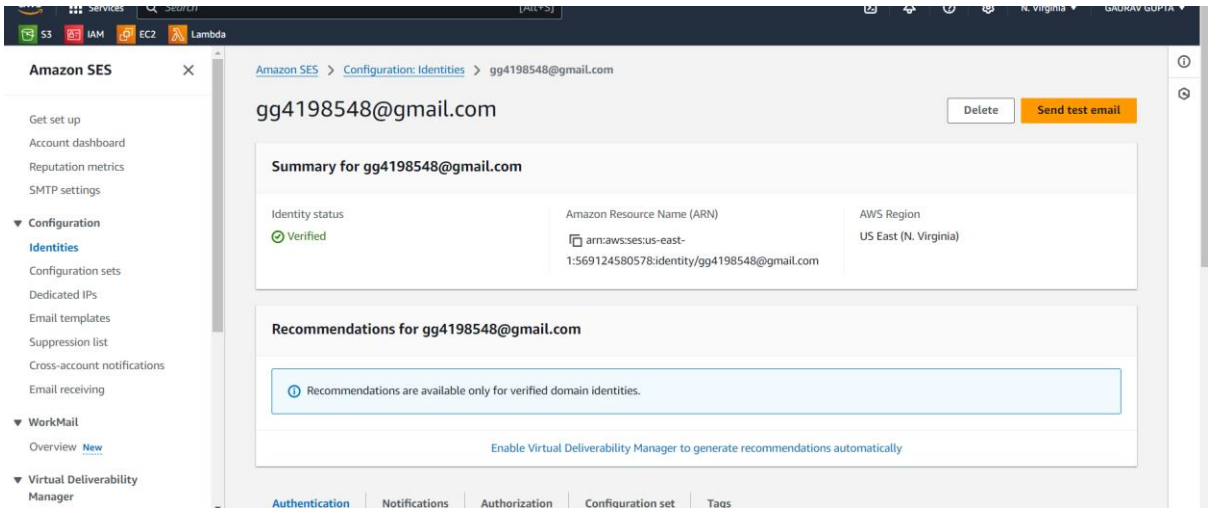
API Gateway


S3
IAM
EC2
Lambda
API Gateway
API Gateway > APIs
APIs (1/1)
Delete
Create API
Find APIs
Name
Description
ID
Protocol
API endpoint type
Created
BellyBrewSubmitHandlerEndpointAPI1
zjrt72u3xh
HTTP
Regional
2024-10-14
APIs
Custom domain names
VPC links
Usage plans
API keys
Client certificates
Settings


API Gateway
APIs
Custom domain names
VPC links
API: BellyBrewSubmit... (zjrt72u3xh)
Develop
Routes
Authorization
Integrations
CORS
Reimport
Export
Deploy
API Gateway > APIs > Routes - BellyBrewSubmitHandlerEndpointAPI1 (zjrt72u3xh)
Routes
Stage: -
Deploy
Routes for BellyBrewSubmitHandlerEndpointAPI1
Create
Search
/submit
POST
Choose a route.

SES


Amazon SES
Get set up
Account dashboard
Reputation metrics
SMTP settings
Configuration
Identities
Configuration sets
Dedicated IPs
Email templates
Suppression list
Cross-account notifications
Email receiving
WorkMail
Overview New
Virtual Deliverability Manager
Amazon SES > Configuration: Identities
Identities
The Identities pane lists your domains, subdomains, and email address identities. All identities must be verified before you use them to send email in Amazon SES. Learn more. The Recommendations pane lists high-impact email authentication issues found for the identities you select and check for recommendations. Learn more
Identities (1) Info
Check for recommendations
Send test email
Delete
Create identity
Search all identities
Identity
Identity type
Identity status
gg4198548@gmail.com
Email address
Verified
Recommendations (0) Info
To see high-impact recommendations, select up to 10 identities in the list above and choose Check for recommendations. If you implement a provided solution, refresh the table by selecting Check for recommendations to ensure it's no longer listed. If you enable Virtual Deliverability Manager, this table will refresh automatically for all your identities.
Search recommendations
Impact
Identity name
Age
Recommendation/Description
Last checked
Resolve issue


Amazon SES
Get set up
Account dashboard
Reputation metrics
SMTP settings
Configuration
Identities
Configuration sets
Dedicated IPs
Email templates
Suppression list
Cross-account notifications
Email receiving
WorkMail
Overview New
Virtual Deliverability Manager
Amazon SES > Configuration: Identities > gg4198548@gmail.com
gg4198548@gmail.com
Delete
Send test email
Summary for gg4198548@gmail.com
Identity status
Verified
Amazon Resource Name (ARN)
arn:aws:ses:us-east-1:569124580578:identity/gg4198548@gmail.com
AWS Region
US East (N. Virginia)
Recommendations for gg4198548@gmail.com
Recommendations are available only for verified domain identities.
Enable Virtual Deliverability Manager to generate recommendations automatically
Authentication
Notifications
Authorization
Configuration set
Tags