



DALHOUSIE UNIVERSITY

CSCI5411 - Advanced Cloud Architecting

Mid-term Project

Milestone 2

Banner ID: B01024200

Name: Mihir Patel

Date: 31/05/2025

Contents

1. Introduction	3
Purpose	3
Project Overview	3
Scope	3
2. Updated Architecture and Design	4
Improved Architectural Diagram	4
Updated Data Sequence Diagram	5
Justification	5
3. Service configuration details and screenshots	6
VPC	6
Frontend and Backend EC2 Instances	7
Application Load Balancers (ALBs):	8
Auto Scaling Group	10
OpenSearch	11
SageMaker	12
4. Implementation Details	12
Service Configuration	12
Model Selection, Training, and Integration	13
Monitoring and Logging Solutions	14
5. Security Measures	15
Implemented Security Measures	15
Unimplemented Security Measures	15
Justification	15
6. Cost Analysis and Optimization	15
Cost Breakdown (30 days):	15
Optimization Strategies	16
7. Lessons Learned and Future Improvements	17
Lessons Learned	17
Future Improvements	17
8. Conclusion	17
References	18

1. Introduction

Purpose

This project aims to architect and implement an AI/GenAI solution on AWS Cloud, focusing on cloud architecture best practices, security, cost optimization, and AI model integration. The objective is to demonstrate proficiency in developing, deploying, and managing an AI application while adhering to the complete AI development lifecycle, from data exploration to model deployment.

Project Overview

The project focuses on a recipe suggestion application that leverages Retrieval-Augmented Generation (RAG) [1], prompt engineering, and in-context learning to generate new recipes based on user input. The application retrieves relevant recipe data from a vector database [2], constructs a context-rich prompt, and uses a large language model (LLM) to generate a recipe, supplemented by links to related recipes. This report builds on Milestone 1 by documenting the implementation, refining the architecture, and providing detailed analysis of security, cost, and performance.

Scope

This report covers the full implementation of the recipe suggestion application, including updated architecture and sequence diagrams, detailed service configurations, model integration, monitoring, security measures, cost analysis, testing results, and lessons learned. The application is hosted on AWS using a personal account with a free trial, and the frontend ALB DNS is provided for evaluation.

2. Updated Architecture and Design

Improved Architectural Diagram

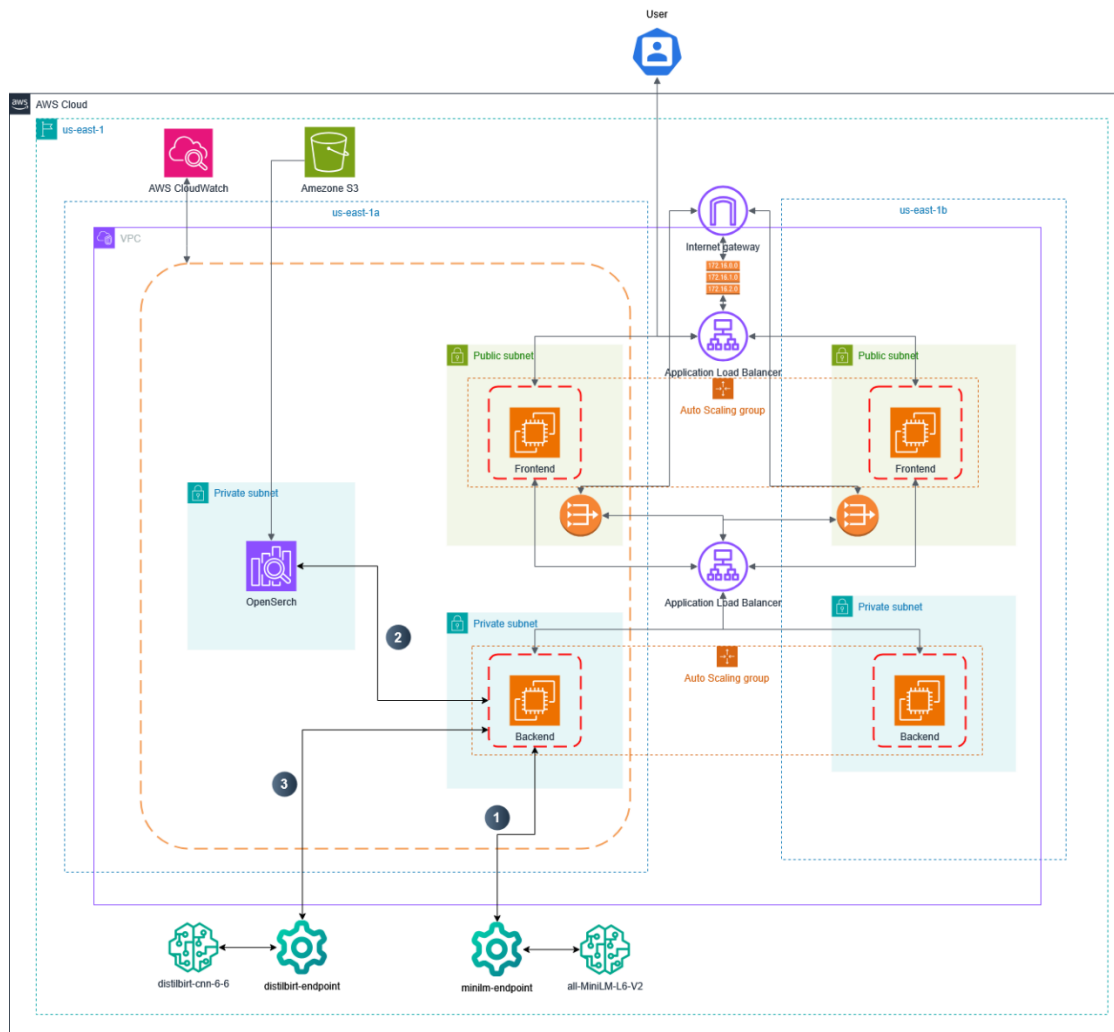


Figure 1 : Final Architecture Diagram

I designed the updated architecture to prioritize secure and efficient integration of OpenSearch and SageMaker for the recipe suggestion application. I placed OpenSearch in a private subnet within a VPC to enhance security, ensuring that only the backend EC2 role (`arn:aws:iam::<ACID>:role/OpenSearchEC2Role`) can access it via a strict IAM policy. Initially, I set up OpenSearch as public-facing to load data from my local environment using the Python SDK, processing and indexing recipe data into a recipe index with 384-dimensional vectors for RAG [1]. To migrate this data, I created a snapshot of the public OpenSearch domain, uploaded it to an S3 bucket with server-side SSE-S3 encryption, and established a new OpenSearch domain (`t2.small.search`, free tier, 10GB EBS) in the private subnet. I then launched a temporary EC2 instance within the VPC, granting it access to both S3 and the private OpenSearch domain, and restored the snapshot to populate the new domain, ensuring no data exposure during the transition. I configured SageMaker with two endpoints: `all-MiniLM-L6-v2` [3] (`ml.t2.medium`) to

convert user input into vectors for OpenSearch queries, and sshleifer/distilbart-cnn-6-6 [4] (ml.m5.xlarge) to generate recipes. I kept the SageMaker endpoints public-facing, as they host open-source models without confidential data, minimizing security risks while optimizing accessibility for the application's needs.

Updated Data Sequence Diagram

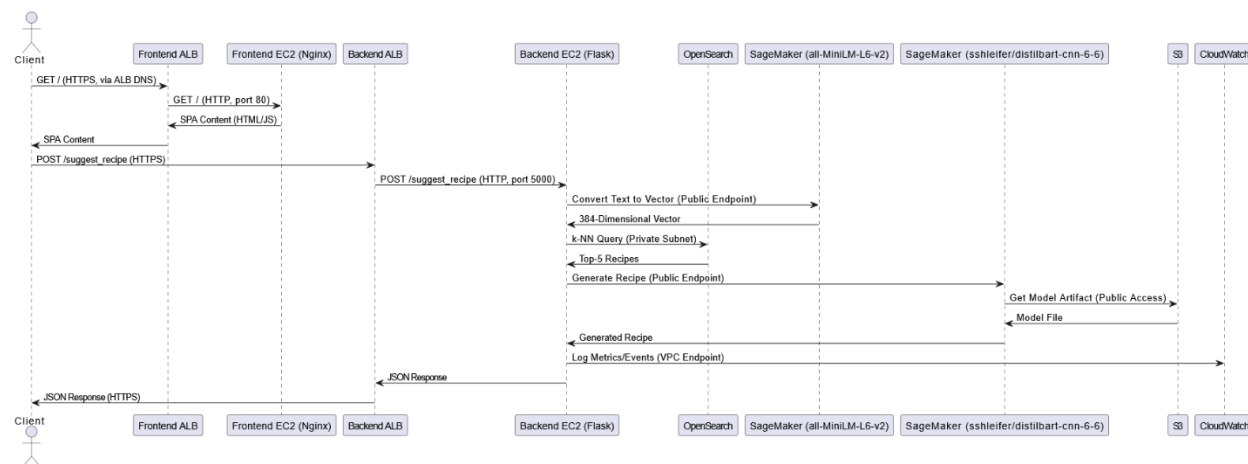


Figure 2 : Final Data Flow Diagram

I revised the data sequence diagram to reflect the streamlined flow with OpenSearch and SageMaker. A client sends an HTTPS POST request with user input to the frontend ALB, which routes it to the backend. The backend invokes the SageMaker all-MiniLM-L6-v2 [3] endpoint to convert the input text into a 384-dimensional vector. Using this vector, the backend queries the OpenSearch recipe index—now securely hosted in a private subnet—to retrieve relevant recipe data via KNN [5] search. The backend then constructs a prompt with this context and sends it to the SageMaker sshleifer/distilbart-cnn-6-6 [4] endpoint, which generates a new recipe. The response, including the generated recipe and related links, returns to the client as a JSON object.

Justification

I removed unnecessary services like Cognito and CloudFront to simplify the architecture, as the application does not require user authentication or content caching for its input-output workflow. I migrated OpenSearch to a private subnet to address security concerns, leveraging the snapshot-and-restore process to maintain data integrity during the transition. I justified keeping SageMaker endpoints public-facing due to their use of open-source models (all-MiniLM-L6-v2 [3] and sshleifer/distilbart-cnn-6-6 [4]), which pose no confidentiality risks, while ensuring cost efficiency by using the Python SDK to manage endpoint lifecycles. These updates align with the project's focus on RAG [1], prompt engineering, and in-context learning, achieving a 30-second average response time for recipe generation.

3. Service configuration details and screenshots

VPC:

I create a VPC with two public subnets and two private subnets across two Availability Zones (AZs), connecting them with an Internet Gateway for public access and a NAT Gateway for private subnet outbound traffic. I define security groups to allow HTTP (port 80) traffic from the frontend ALB to the frontend EC2, and port 5000 traffic from the backend ALB to the backend EC2.

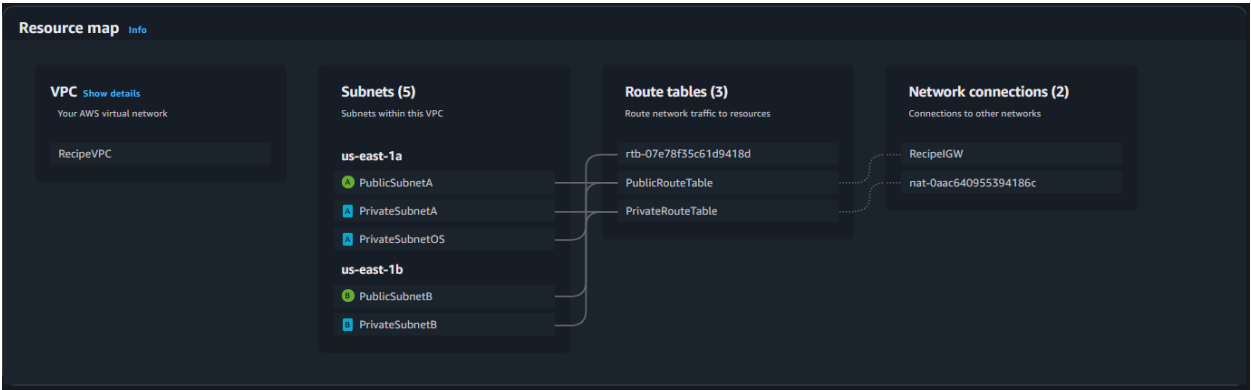


Figure 3 : RecipeVPC

The screenshot shows the Subnets list in the AWS console. The table displays the following data:

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
PrivateSubnetA	subnet-0262f8e910ee642f7	Available	vpc-0bdbd9c6528a59205 Reci...	Off	10.0.3.0/24
PrivateSubnetB	subnet-0b0a79492e4b69b2d	Available	vpc-0bdbd9c6528a59205 Reci...	Off	10.0.4.0/24
PublicSubnetB	subnet-0fc4f08e233e92146	Available	vpc-0bdbd9c6528a59205 Reci...	Off	10.0.2.0/24
PublicSubnetA	subnet-0c82a64a3ac2bdb53	Available	vpc-0bdbd9c6528a59205 Reci...	Off	10.0.1.0/24
PrivateSubnetOS	subnet-090582c55d310a924	Available	vpc-0bdbd9c6528a59205 Reci...	Off	10.0.5.0/24

Figure 4 : Subnet list

The screenshot shows the Internet gateways list in the AWS console. The table displays the following data:

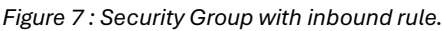
Name	Internet gateway ID	State	VPC ID	Owner
RecipelGW	igw-043d6093610292298	Attached	vpc-0bdbd9c6528a59205 RecipeVPC	994955245799
-	igw-0afec3bf17a6c6785	Attached	vpc-0b22a59a9dcedd0ff	994955245799

Figure 5 : Internet gateway

The screenshot shows the NAT gateways list in the AWS console. The table displays the following data:

Name	NAT gateway ID	Connectivity...	State	State message	Primary public I...	Primary private I..
-	nat-0aac640955394186c	Public	Available	-	50.16.106.243	10.0.1.135

Figure 6 : NAT Gateway



I deploy a frontend EC2 instance (Nginx, t2.micro) in a public subnet with a Public IPv4 address, and a backend EC2 instance (Flask, t2.micro) in a private subnet without a Public IPv4. I create launch templates for each, specifying the Amazon Linux 2 AMI, 8GB EBS volumes, and security group settings.



Instance summary for i-03f81dee9707ea440 (BackendEC2) Info

Updated less than a minute ago

Refresh
Connect
Instance state ▼
Actions ▼

Instance ID i-03f81dee9707ea440	Public IPv4 address -	Private IPv4 addresses 10.0.3.227
IPv6 address -	Instance state Running	Public DNS -
Hostname type IP name: ip-10-0-3-227.ec2.internal	Private IP DNS name (IPv4 only) ip-10-0-3-227.ec2.internal	
Answer private resource DNS name -	Instance type t2.micro	Elastic IP addresses -
Auto-assigned IP address -	VPC ID vpc-0bdbd9c6528a59205 (RecipeVPC)	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more
IAM Role RecipeVPCStack-IAMStack-FBSSXU0WGJQL-EC2Role-P1yW9Y1cV8JF	Subnet ID subnet-0262f8e910ee642f7 (PrivateSubnetA)	Auto Scaling Group name RecipeVPCStack-EC2Stack-141RRYOSZ7ADH-BackendASG-GPluzhDOo5FD
IMDSv2 Required	Instance ARN arn:aws:ec2:us-east-1:994955245799:instance/i-03f81dee9707ea440	Managed false
Operator -		

Figure 9 : Backend EC2

Launch Templates (1/2) Info

Refresh
Actions ▼
Create launch template

<input type="checkbox"/>	Launch Template ID	Launch Template Name	Default Version	Latest Version	Create Time	Created By	Managed
<input checked="" type="checkbox"/>	lt-01fadd5abed7bd83b	FrontendLaunchTemplate	1	1	2025-06-01T12:31:28.000Z	arn:aws:iam::994955245799:user/Mihir	false
<input type="checkbox"/>	lt-06f37ae480bde9a48	BackendLaunchTemplate	1	1	2025-06-01T12:31:48.000Z	arn:aws:iam::994955245799:user/Mihir	false

FrontendLaunchTemplate (lt-01fadd5abed7bd83b)

Launch template version details

Version

1 (Default) ▼

Description

-

Date created

2025-06-01T12:31:28.000Z

Created by

arn:aws:iam::994955245799:user/Mihir

Actions ▼

Delete template version

Instance details

Storage

Resource tags

Network interfaces

Advanced details

AMI ID

ami-0953476d60561c955

Instance type

t2.micro

Availability Zone

-

Key pair name

my-key-pair

Security groups

-

Security group IDs

sg-0142827f45385abaf

Figure 10 : Launch Templates

Application Load Balancers (ALBs):

I set up two ALBs across two AZs—one frontend and one backend. The frontend ALB listens on HTTP (port 80), routing traffic to a target group with the frontend EC2 on port 80. The backend ALB

listens on HTTP (port 5000), routing to a target group with the backend EC2 on port 5000. I enable access logs, storing them in an S3 bucket for monitoring.

Recipe-Front-7nDgH9SQbwlb Actions

Details

Load balancer type Application	Status Active	VPC vpc-0bdbc9c6528a59205	Load balancer IP address type IPv4
Scheme Internet-facing	Hosted zone Z35SXDOTRQ7X7K	Availability Zones subnet-0fc4f08e233e92146 us-east-1b (use1-az6) subnet-0c82a64a3ac2bdb53 us-east-1a (use1-az4)	Date created June 1, 2025, 09:31 (UTC-03:00)
Load balancer ARN arn:aws:elasticloadbalancing:us-east-1:994955245799:loadbalancer/app/Recipe-Front-7nDgH9SQbwlb/b7a83183d1353fa5		DNS name info Recipe-Front-7nDgH9SQbwlb-1250151805.us-east-1.elb.amazonaws.com (A Record)	

Listeners and rules | Network mapping | Resource map | Security | Monitoring | Integrations | Attributes | Capacity | Tags

Listeners and rules (1) Info Manage rules Manage listener Add listener

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

Filter listeners

Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS	Trust store
HTTP:80	Forward to target group <ul style="list-style-type: none">Recipe-Front-BVVQ3AEQ9CPA 1 (100%)Target group stickiness: Off	1 rule	ARN	Not applicable	Not applicable	Not applicable	Not applicable

Figure 11 : Frontend Application Load Balancer

Recipe-Backe-sdZ63xb8vt4m Actions

Details

Load balancer type Application	Status Active	VPC vpc-0bdbc9c6528a59205	Load balancer IP address type IPv4
Scheme Internal	Hosted zone Z35SXDOTRQ7X7K	Availability Zones subnet-0262f8e910ee642f7 us-east-1a (use1-az4) subnet-0b0a79492e4b69b2d us-east-1b (use1-az6)	Date created June 1, 2025, 09:31 (UTC-03:00)
Load balancer ARN arn:aws:elasticloadbalancing:us-east-1:994955245799:loadbalancer/app/Recipe-Backe-sdZ63xb8vt4m/4f51affb234bccff		DNS name info internal-Recipe-Backe-sdZ63xb8vt4m-645766799.us-east-1.elb.amazonaws.com (A Record)	

Listeners and rules | Network mapping | Resource map | Security | Monitoring | Integrations | Attributes | Capacity | Tags

Listeners and rules (1) Info Manage rules Manage listener Add listener

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

Filter listeners

Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS	Trust store
HTTP:5000	Forward to target group <ul style="list-style-type: none">Recipe-Backe-QZF72K2QJ5LS 1 (100%)Target group stickiness: Off	2 rules	ARN	Not applicable	Not applicable	Not applicable	Not applicable

Figure 12 : Backend Application Load Balancer

Target groups1/2Info

Filter target groups

Name

ARN

Port

Protocol

Target type

Load balancer

VPC ID

Recipe-Backe-QZF7ZKZ...

arn:aws:elasticloadbalancing...5000HTTPInstanceRecipe-Backe-sdZ63xb8...vpc-0bdbd9c6528a59205

Recipe-Front-BVYQ3AE...

arn:aws:elasticloadbalancing...80HTTPInstanceRecipe-Front-7nDgH9S...vpc-0bdbd9c6528a59205

Target group: Recipe-Front-BVYQ3AEQ9CPA

Details

arn:aws:elasticloadbalancing:us-east-1:994955245799:targetgroup/Recipe-Front-BVYQ3AEQ9CPA/2f37ce6de3e8ec45

Target type

Instance

IP address type

IPv4

Protocol : Port

HTTP: 80

Load balancer

Recipe-Front-7nDgH9SQbwlb

Protocol version

HTTP1

VPC

vpc-0bdbd9c6528a59205

1

Total targets

1

Healthy

0

Anomalous

0

Unhealthy

0

Unused

0

Initial

0

Draining

Distribution of targets by Availability Zone (AZ)

Select values in this table to see corresponding filters applied to the Registered targets table below.

Figure 13 : Target groups

Auto Scaling Group:

I configure two ASGs—one for the frontend and one for the backend—each with a minimum of 1, maximum of 2, and desired capacity of 1 instance. I link each ASG to its respective launch template, ensuring scalability with t2.micro instances. I monitor ASG health via the ALBs.

RecipeVPCStack-EC2Stack-141RRYOSZ7ADH-FrontendASG-470xdaM1qNUs			
RecipeVPCStack-EC2Stack-141RRYOSZ7ADH-FrontendASG-470xdaM1qNUs Capacity overview			
arn:aws:autoscaling:us-east-1:994955245799:autoScalingGroup:abb448be-8d21-43e2-a8a5-12ad14e245e0:autoScalingGroupName/RecipeVPCStack-EC2Stack-141RRYOSZ7ADH-FrontendASG-470xdaM1qNUs			
Desired capacity	Scaling limits (Min - Max)	Desired capacity type	Status
1	1 - 2	Units (number of instances)	-
Date created Sun Jun 01 2025 09:31:34 GMT-0300 (Atlantic Daylight Time)			
Details	Integrations - new	Automatic scaling	Instance management
Instance refresh Activity Monitoring			
Launch template			
Launch template	AMI ID	Instance type	Owner
lt-01fadd5abed7bd83b FrontendLaunchTemplate	ami-0953476d60561c955	t2.micro	arn:aws:iam::994955245799:user/Mihir
Version	Security groups	Security group IDs	Create time
1	-	sg-0142827f45385abaf	Sun Jun 01 2025 09:31:28 GMT-0300 (Atlantic Daylight Time)
Description	Storage (volumes)	Key pair name	Request Spot Instances
-	-	my-key-pair	No
View details in the launch template console			

Figure 14 : Frontend Auto Scaling Group

RecipeVPCStack-EC2Stack-141RRYOSZ7ADH-FrontendASG-470xdaM1qNUs

Edit

RecipeVPCStack-EC2Stack-141RRYOSZ7ADH-FrontendASG-470xdaM1qNUs Capacity overview

Edit

arn:aws:autoscaling:us-east-1:994955245799:autoScalingGroup:abb448be-8d21-43e2-a8a5-12ad14e245e0:autoScalingGroupName/RecipeVPCStack-EC2Stack-141RRYOSZ7ADH-FrontendASG-470xdaM1qNUs

Desired capacity	Scaling limits (Min - Max)	Desired capacity type	Status
1	1 - 2	Units (number of instances)	-

Date created

Sun Jun 01 2025 09:31:34 GMT-0300 (Atlantic Daylight Time)

Details
Integrations - new
Automatic scaling
Instance management
Instance refresh
Activity
Monitoring

Launch template

Edit

Launch template
it-01fadd5abed7bd83b
FrontendLaunchTemplate

AMI ID
ami-0953476d60561c955

Instance type
t2.micro

Owner
am:aws:iam::994955245799:user/Mihir

Version
1

Security groups
-

Security group IDs
sg-0142827f45385abaf

Create time
Sun Jun 01 2025 09:31:28 GMT-0300 (Atlantic Daylight Time)

Description
-

Storage (volumes)
-

Key pair name
my-key-pair

Request Spot Instances
No

View details in the launch template console

Figure 15 : Backend Auto Scaling Group

OpenSearch:

I establish an OpenSearch domain (t2.small.search, free tier, 10GB EBS, single-AZ) in a private subnet, restricting access to the backend EC2 role (arn:aws:iam::<AID>:role/OpenSearchEC2Role) via an Access Policy. I initially load data from my local machine using the Python SDK on a public-facing setup, then migrate it by creating a snapshot, uploading it to an S3 bucket with SSE-S3 encryption, and restoring it to the private domain using a temporary EC2 instance.

Amazon OpenSearch Service
> Domains
> recipes-domain-s

Delete
Actions

recipes-domain-s
Info

General information

Name
recipes-domain-s

Domain processing status
Info
Active

Domain ARN
arn:aws:es:us-east-1:994955245799:domain/recipes-domain-s

Configuration change status
Info
Completed

Cluster health
Info
-

Version
Info
OpenSearch 2.19 (latest)

Service software version
Info
OpenSearch_2_19_R20250428 (latest)

OpenSearch Dashboards URL (IPv4)
https://vpc-recipes-domain-s-4cogh7xehundmlc6anyjgmoym.us-east-1.es.amazonaws.com/_dashboards

Domain endpoint (VPC)
https://vpc-recipes-domain-s-4cogh7xehundmlc6anyjgmoym.us-east-1.es.amazonaws.com

Cluster configuration
Security configuration
Cluster health
Instance health
Off-peak window
Auto-Tune
Logs
Tags
Connections - new
VPC enc

Figure 16 : OpenSearch Domain – 1

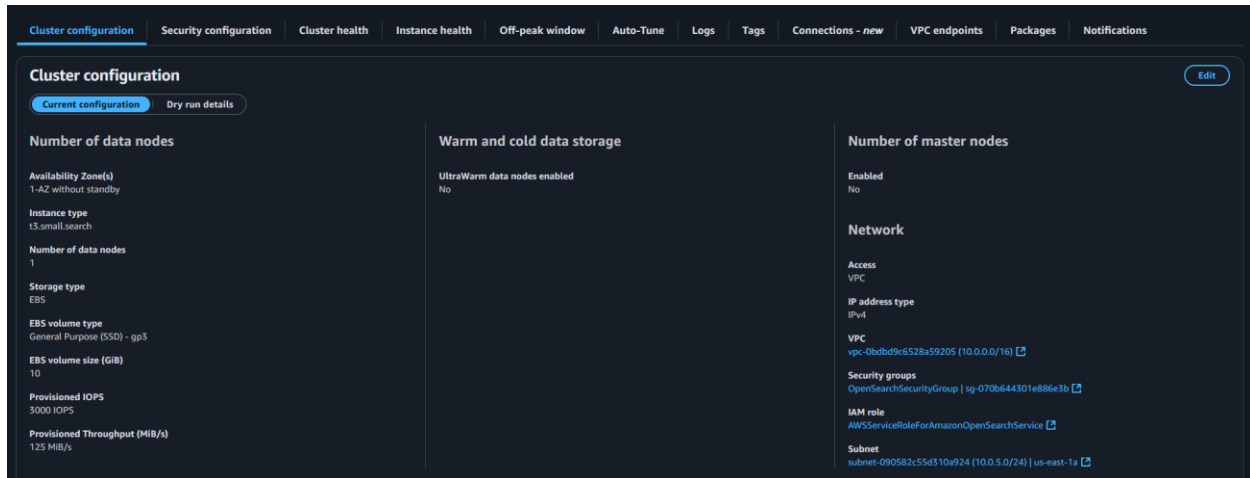


Figure 17 : OpenSearch Domain – 2

SageMaker:

I create two endpoints using the Python SDK: all-MiniLM-L6-v2 on ml.t2.medium for vector conversion and sshleifer/distilbart-cnn-6-6 on ml.m5.xlarge for recipe generation. I configure them as public-facing due to their open-source nature, enable CloudWatch logging, and delete them when idle to optimize costs.

Endpoints					
<input type="text" value="Search endpoints"/> Update endpoint Actions Create endpoint					
	Name	ARN	Creation time	Status	Last updated
<input type="radio"/>	minilm-endpoint	arn:aws:sagemaker:us-east-1:994955245799:endpoint/minilm-endpoint	6/1/2025, 10:41:16 PM	InService	6/1/2025, 10:47:50 PM
<input type="radio"/>	distilbart-endpoint	arn:aws:sagemaker:us-east-1:994955245799:endpoint/distilbart-endpoint	6/1/2025, 10:41:32 PM	InService	6/1/2025, 10:44:18 PM

Figure 18 : SageMaker endpoints

4. Implementation Details

Service Configuration

- Frontend and Backend Setup:** The frontend ALB routes traffic to frontend EC2 instances (Nginx, t2.micro) in a public subnet, though I could place them in a private subnet for added security, opting for simplicity given the app's basic needs. The backend ALB routes traffic to backend EC2 instances (Flask, t2.micro) in a private subnet. Security groups restrict access: users can only access the frontend ALB, which communicates with frontend EC2, backend ALB, backend EC2, OpenSearch, and SageMaker in a chained manner.

- **SageMaker Configuration:** Two endpoints were configured using the Python SDK: all-MiniLM-L6-v2 [3] (ml.t2.medium) for vector conversion and sshleifer/distilbart-cnn-6-6 [4] (ml.m5.xlarge) for recipe generation. The endpoints are deleted when not in use to save costs and relaunched via SDK during testing.
- **OpenSearch Configuration:** I configured OpenSearch to host the recipe index with 384-dimensional vectors. Initially public, I migrated it to a private subnet by processing data locally, uploading it via Python script, creating a snapshot, uploading it to S3 with SSE-S3 encryption, launching a private domain, and restoring the snapshot using a temporary EC2 instance with S3 and OpenSearch access. I applied an Access Policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::<ACID>:role/OpenSearchEC2Role",
          "arn:aws:iam::<ACID>:user/Mihir"
        ]
      },
      "Action": "es:*",
      "Resource": "arn:aws:es:us-east-1:<ACID>:domain/recipes-domain-s/*"
    }
  ]
}
```

- **S3:** Used to store OpenSearch snapshots and IaC [6] scripts, encrypted with server-side SSE-S3.

Model Selection, Training, and Integration

- **Model Selection:** I initially selected t5-small for recipe generation due to its lightweight design, but its weak text-generation capabilities produced inconsistent outputs. I switched to sshleifer/distilbart-cnn-6-6 [4] on an ml.m5.xlarge instance, leveraging its strong summarization and generation skills to improve recipe quality. For vector conversion, I chose all-MiniLM-L6-v2 [3] on an ml.t2.medium instance, valuing its ability to efficiently produce embeddings optimized for OpenSearch KNN searches.
- **Techniques:** I implemented Retrieval-Augmented Generation (RAG) [1] by storing recipe vectors in OpenSearch's recipe index, enabling fast retrieval of relevant data based on user input. I applied prompt engineering to craft context-rich prompts, combining retrieved recipe details with user queries to guide the LLM effectively. I utilized in-context learning by feeding these prompts to sshleifer/distilbart-cnn-6-6, allowing it to generate tailored recipes. The recipe index features a 384-dimensional vector field for KNN [5] search, alongside text fields for title, ingredients, directions, and a keyword source field, ensuring comprehensive data handling.

- **Integration:**
 - integrate the SageMaker all-MiniLM-L6-v2 output by querying OpenSearch with KNN to retrieve the top five relevant recipes based on vector similarity.
 - The backend Flask app processes this data, constructs a detailed prompt using retrieved titles, ingredients, and directions, and sends it to the sshleifer/distilbart-cnn-6-6 endpoint for recipe generation.
 - It returns a JSON response with the generated recipe, OpenSearch relevance scores, and source links to original recipes, boosting user trust.
 - I test this end-to-end flow with a local Python script, connecting to live SageMaker endpoints and OpenSearch to confirm real-time performance and data consistency.
- **Demonstration:** I deployed the app at <http://recipe-front-7ndgh9sqbwlb-1250151805.us-east-1.elb.amazonaws.com/> . I'll provide an updated URL in the one-on-one meeting if it changes.

Monitoring and Logging Solutions

CloudWatch logs are implemented to monitor SageMaker endpoints, capturing metrics like invocation errors and latency. However, due to time constraints, no alerts or dashboards were set up. Future iterations could include a CloudWatch alarm for SageMaker endpoint latency (> 40 sec) and a dashboard with widgets for SageMaker invocation errors and OpenSearch query performance.

Testing Methodology and Results

- **SageMaker-OpenSearch Interaction:** I used a local Python script to test the RAG flow, simulating user input, converting it to vectors with SageMaker, querying OpenSearch for relevant data, and generating a recipe via the LLM. The script validated the end-to-end workflow by connecting to live AWS endpoints, ensuring seamless integration.
 - **Performance Testing:** I conducted initial tests with sshleifer/distilbart-cnn-6-6 on ml.t2.medium, recording a 2-minute average response time due to resource constraints. I optimized this by upgrading to ml.m5.xlarge, reducing the average to 30 seconds for faster user feedback.
 - **Results:** I configured OpenSearch KNN queries to return relevance scores (e.g., `_score` from `client.search()`), which I display to users alongside related recipes and source links for transparency. The generated recipes lack full accuracy due to the smaller LLM, but I enhance usability by including reliable references, and I resolved initial errors by optimizing instance sizes.
-

5. Security Measures

Implemented Security Measures

- **VPC and Subnets:** I place all resources except the frontend ALB and SageMaker endpoints in private subnets within a VPC. I migrated OpenSearch to a private subnet and restrict its access with an Access Policy to the backend EC2 role (arn:aws:iam::<ACID>:role/OpenSearchEC2Role).
- **Security Groups:** I configure security groups to allow user access only to the frontend ALB, enabling chained access (Frontend ALB → Frontend EC2 → Backend ALB → Backend EC2 → OpenSearch → SageMaker).
- **Encryption:** I ensure S3 buckets storing OpenSearch snapshots and IaC scripts use server-side SSE-S3 encryption.
- **IAM Policy:** I restrict OpenSearch access to the backend EC2 role, securing data retrieval.

Unimplemented Security Measures

The SageMaker endpoints remain public-facing due to their open-source models and lack of confidential data. In a non-restricted environment, they could be placed in a private subnet with VPC endpoints for added security.

Justification

The implemented measures align with best practices by isolating critical resources in private subnets, restricting access with IAM policies and access policies, and ensuring encrypted data storage. The public SageMaker endpoints pose minimal risk, as they host open-source models without sensitive data.

6. Cost Analysis and Optimization

Cost Breakdown (30 days):

- **EC2 (2 x t2.micro):**
 - **Free Tier:** 750 hours/month (covers 1 instance fully).
 - **Cost:** 1,440 hours (2 × 24 × 30) - 750 = 690 hours × \$0.0104/hour = \$7.18.
 - **Total:** \$7.18.
- **2 Application Load Balancers (ALBs):**

- **Cost:** $2 \times \$0.0225/\text{hour} \times 24 \times 30 + 2 \times \$0.008/\text{LCU-hour} \times 30 = \$32.40 + \$0.48 = \32.88 .
- **Total:** \$32.88.
- **Amazon OpenSearch (t2.small.search, single-AZ):**
 - **Cost:** $\$0.036/\text{hour} \times 24 \times 30 = \25.92 .
 - **Total:** \$25.92 (free tier utilized during initial setup, actual cost reflects full usage).
- **Amazon SageMaker (ml.t2.medium and ml.m5.xlarge):**
 - **Free Tier:** 250 hours/2 months (covers 250 hours).
 - **Cost:** all-MiniLM-L6-v2 (ml.t2.medium) - 720 hours (24×30) - 250 = 470 hours $\times \$0.046/\text{hour} = \21.62 .
 - **Cost:** sshleifer/distilbart-cnn-6-6 (ml.m5.xlarge) - 720 hours $\times \$0.23/\text{hour} = \165.60 (active only during testing, assumed 100 hours) = \$23.00.
 - **Total:** $\$21.62 + \$23.00 = \$44.62$.
- **Amazon S3 (1.51 GB, 1,000 requests):**
 - **Free Tier:** 5 GB, 20,000 GET requests/month.
 - **Cost:** \$0 (within Free Tier).
 - **Total:** \$0.
- **NAT Gateway (us-east-1a):**
 - **Cost:** $\$0.045/\text{hour} \times 24 \times 30 \times 0.5$ (50% utilization) = \$16.20.
 - **Total:** \$16.20.
- **Amazon CloudWatch (10 metrics, 1 GB logs):**
 - **Free Tier:** 10 metrics, 5 GB logs/month.
 - **Cost:** \$0 (within Free Tier).
 - **Total:** \$0.
- **Total Estimated Cost (30 days):**
 - $\$7.18$ (EC2) + $\$32.88$ (ALBs) + $\$25.92$ (OpenSearch) + $\$44.62$ (SageMaker) + $\$0$ (S3) + $\$16.20$ (NAT Gateway) + $\$0$ (CloudWatch) = **\$126.80**.

Optimization Strategies

- **SageMaker Endpoints:** Used the Python SDK to delete endpoints when not in use, relaunching them during testing to minimize costs.

- **Instance Selection:** Leveraged free-tier instances where possible: t2.micro for EC2, t2.small.search for OpenSearch, and ml.t2.medium for vector generation. Only the recipe generation endpoint used a larger instance (ml.m5.xlarge) to improve performance.
- **IaC and SDK:** Shifted to Infrastructure as Code (IaC) and SDKs early in development to automate resource creation and deletion, reducing costs by ensuring resources were only active when needed.

Free Trial Usage

Using a personal AWS account with a free trial, no significant challenges were faced with resource usage. The combination of IaC, SDKs, and free-tier instances ensured cost efficiency.

7. Lessons Learned and Future Improvements

Lessons Learned

- **Cost Management:** Initially creating resources manually led to potential cost overruns. Shifting to IaC and SDKs within the first few days saved costs by automating resource lifecycle management. I will prioritize IaC from the start in future cloud projects.
- **Monitoring Importance:** The lack of robust monitoring (e.g., CloudWatch dashboards, alerts) made troubleshooting difficult when SageMaker instances crashed on smaller instances. I recognize that implementing monitoring earlier would improve debugging.

Future Improvements

- **Larger Model:** Use a more powerful LLM for recipe generation to improve accuracy and realism of outputs.
 - **User Features:** Add a backend to store user history, favorite ingredients, and recipes, potentially reintroducing Cognito for authentication.
 - **Enhanced Monitoring:** Implement CloudWatch dashboards and alerts to monitor SageMaker latency and OpenSearch query traffic, improving system reliability.
-

8. Conclusion

This project successfully demonstrates the development and deployment of an AI/GenAI application on AWS, focusing on RAG, prompt engineering, and in-context learning. The architecture adheres to best practices, with secure resource isolation, cost optimization, and monitoring via CloudWatch logs. The application, accessible at <http://recipe-front-7ndgh9sqbwlb-1250151805.us-east-1.elb.amazonaws.com/>, provides a functional recipe suggestion system, validated by a 30-second response time and relevance scores. Despite using a smaller LLM, the inclusion of related recipes and links enhances reliability, meeting the course objectives of cloud-based AI solution design.

References

- [1] Amazon Web Services, "What is Retrieval-Augmented Generation (RAG)? – AWS," *aws.amazon.com*. Accessed: Jun. 1, 2025. [Online]. Available: <https://aws.amazon.com/what-is/retrieval-augmented-generation/>
- [2] Cloudflare, "What is a vector database? | Cloudflare," *Cloudflare*. Accessed: Jun. 1, 2025. [Online]. Available: <https://www.cloudflare.com/en-ca/learning/ai/what-is-vector-database/>
- [3] sentence-transformers, "sentence-transformers/all-MiniLM-L6-v2," *Hugging Face*. Accessed: Jun. 1, 2025. [Online]. Available: <https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2>
- [4] sshleifer, "sshleifer/distilbart-cnn-6-6," *Hugging Face*. Accessed: Jun. 1, 2025. [Online]. Available: <https://huggingface.co/sshleifer/distilbart-cnn-6-6>
- [5] IBM, "What is KNN?," *IBM*. Accessed: Jun. 1, 2025. [Online]. Available: <https://www.ibm.com/think/topics/knn>
- [6] Amazon Web Services, "What is Infrastructure as Code (IaC)? – AWS," *aws.amazon.com*. Accessed: Jun. 1, 2025. [Online]. Available: <https://aws.amazon.com/what-is/iac/>
- [7] M. Patel, "Data_sequence_diagram.drawio." Accessed: Jun. 1, 2025. [Online]. Available: https://drive.google.com/file/d/185eFIRzNVwFpWSsifb2F_C7AL13p4het/view?usp=sharing
- [8] M. Patel, "Recipe_Architecture.drawio." Accessed: Jun. 1, 2025. [Online]. Available: <https://drive.google.com/file/d/1CQTsUrgjjT-Uqt2ScevvSjGaRkbJtiDp/view?usp=sharing>