Link to the demo: https://dc8eru1q29n2p.cloudfront.net

1. Domain

The chosen domain for this project is Education and personalised learning, more specifically, it is an AI-Assisted Code Understanding. Sometimes the code files I am looking at for open-source development are very complex to understand because of the use of libraries that I might not have used or seen before, or because of poor code writing practises. The objective of making this tool is to find or analyse the exact location in a file where a certain functionality exists, and answer code related questions.

2. Problem Statement

Modern software teams deal with large, complex codebases where understanding legacy code, onboarding new developers, or finding relevant functionality is time-consuming and error-prone. Traditional keyword-based search tools (like grep, or built-in IDE search) fail to capture the semantic meaning of code, leading to poor discovery, redundant work, and slow development cycles.

3. Business Case

Organizations waste hundreds of hours per developer annually on tasks like:

- Understanding unfamiliar code sections
- Searching for functions, APIs, or logic
- Asking teammates questions that could be answered from code.

This results in:

- Delayed feature delivery
- Frustrated engineers
- Higher onboarding time
- Increased bugs due to misunderstood logic.

The proposed system enables:

- Uploading a codebase
- Embedding it semantically using **CodeBERT**
- Creating an efficient vector index with **FAISS**
- Letting users ask **natural-language questions**
- Returning accurate, contextually relevant answers via model deployed on **SageMaker.**

4. Functional Requirements

- The system should allow users to upload either one zip file or a Github link to a codebase.
- The system should process the files so that when the user asks questions related to the content in the files, the system can answer the questions.
- The system should return answers that are relevant to the content of the uploaded code.
- The system should allow users to ask natural-language questions about their uploaded codebase.
- The received response from the system must be in English and should be visible on screen.
- A user should be able to ask multiple questions after they upload files or the Github link.
- The system should notify users if there is an error during upload, processing, or querying.

5. Non-Functional Requirements

Performance:

- The system should not take more than 10 minutes to answer user queries.
- The system should not take more than 10 minutes to process the uploaded files.
- The system should support more than one user concurrently.

Scalability:

- The system should be able to spawn more compute power (EC2 instances) if the demand spikes so that system doesn’t become slow or throw error when it receives multiple requests at the same time.

Security:

- The system should restrict access to user-uploaded data such that only the owning user can query their code.
- All data transfers between client and server shall be encrypted.
- Back-end functionalities should only be accessible from the front-end and nowhere else (like terminal or other tools).
- Only the front-end / website / code should be able to access the back-end processing functionalities.

Data Handling:

- Data should be deleted after 30 days unless compliance to a government policy requires the storage of data for longer periods, or there is an explicit business requirement.

Monitoring and Logging:

- Requests to the Lambda functions and other AWS services must be monitored through CloudWatch.
- The system should expose operational metrics (e.g., job queue length, task duration, query latency) for observability.

Availability and Reliability:

- The system should be available most of the time to take user input.

6. Architecture Diagram

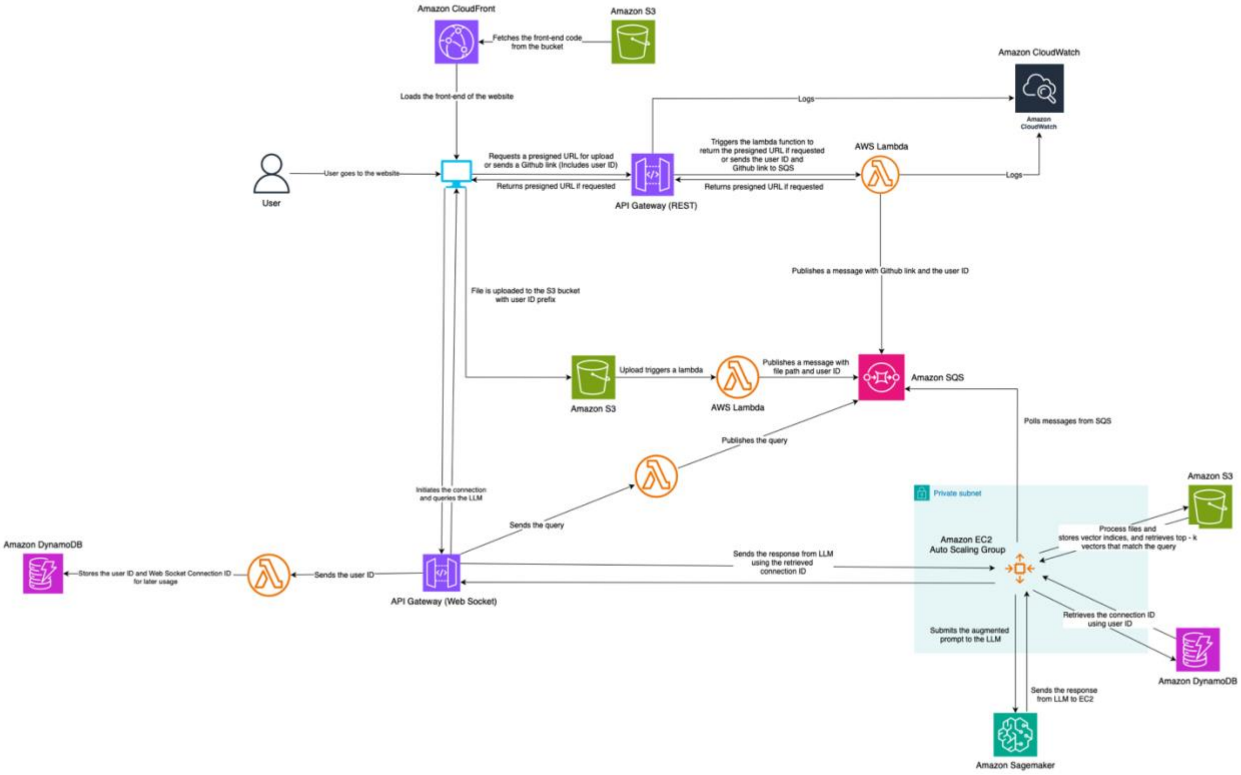

Amazon CloudFront
Amazon S3
Fetches the front-end code from the bucket
Loads the front-end of the website
Amazon CloudWatch
Logs
Amazon CloudWatch
User
User goes to the website
Requests a presigned URL for upload or sends a Github link (Includes user ID)
Returns presigned URL if requested
API Gateway (REST)
Triggers the lambda function to return the presigned URL if requested or sends the user ID and Github link to SQS
Returns presigned URL if requested
AWS Lambda
Logs
Publishes a message with Github link and the user ID
File is uploaded to the S3 bucket with user ID prefix
Amazon S3
Upload triggers a lambda
AWS Lambda
Publishes a message with file path and user ID
Amazon SQS
Polls messages from SQS
Publishes the query
Initiates the connection and queries the LLM
Sends the query
Amazon DynamoDB
Stores the user ID and Web Socket Connection ID for later usage
Sends the user ID
API Gateway (Web Socket)
Sends the response from LLM using the retrieved connection ID
Private subnet
Amazon EC2 Auto Scaling Group
Amazon S3
Process files and stores vector indices, and retrieves top - k vectors that match the query
Retrieves the connection ID using user ID
Amazon DynamoDB
Submits the augmented prompt to the LLM
Sends the response from LLM to EC2
Amazon Sagemaker

Fig. 6.1 Architecture Diagram

I. User first access the website through Amazon CloudFront.

II. When the user lands on the website, the browser initiates the connection to web socket so that the responses could be retrieved later on.

III. Once the connection is open, a unique user ID is generated and wrapped in a message. This message is sent to the web socket.

IV. This causes a lambda function to trigger. The lambda function will then extract the user ID from the message, and the connection ID from its invocation metadata, and stores them in the DynamoDB.

V. Now user either uploads the zip file of the code, or enters the corresponding Github URL, and sends a request to API Getway.

VI. API Gateway triggers a lambda function that will process these inputs. If the user has uploaded zip file of the code, it will generate a presigned URL and send that URL to the front-end. If the user has uploaded Github URL, the URL along with the user ID will be passed to the Amazon SQS. Once the user uploads the file using the presigned URL, the file path will be passed to the same Amazon SQS.

VII. EC2 instance managed by Elastic Beanstalk will poll the messages from the queue and start processing the uploaded files. It will embed the files in chunks using CodeBERT model and store the vectors in FAISS index, the metadata in another S3 bucket.

VIII. Once the processing is finished, it will query the DynamoDB for the connection ID using the user ID to notify the front-end using the web socket.

IX. The front-end will allow user to input query.

X. Once the query is entered, it will be sent as a message along with the user ID to the web socket, which will trigger a Lambda function.

XI. The lambda function will put this message in a queue. EC2 will poll the message and starts processing the query.

XII. The query will be embedded using CodeBERT. Then the program will retrieve vectors stored in FAISS index and search for the top – k similar embeddings / vectors that match the query vector.

XIII. Once the matches have been found, they will be augmented into the user prompt and passed to the AI model deployed (Code Llama / Meta Llama / CodeBERT) on Amazon Sagemaker.

XIV. Amazon Sagemaker will make use of augmented context into the prompt and return a response to the EC2 instance.

XV. EC2 instance will send the response in chunks to the front-end of the website through the web socket.

XVI. The user will be able to see the responses as they are received from the back-end.

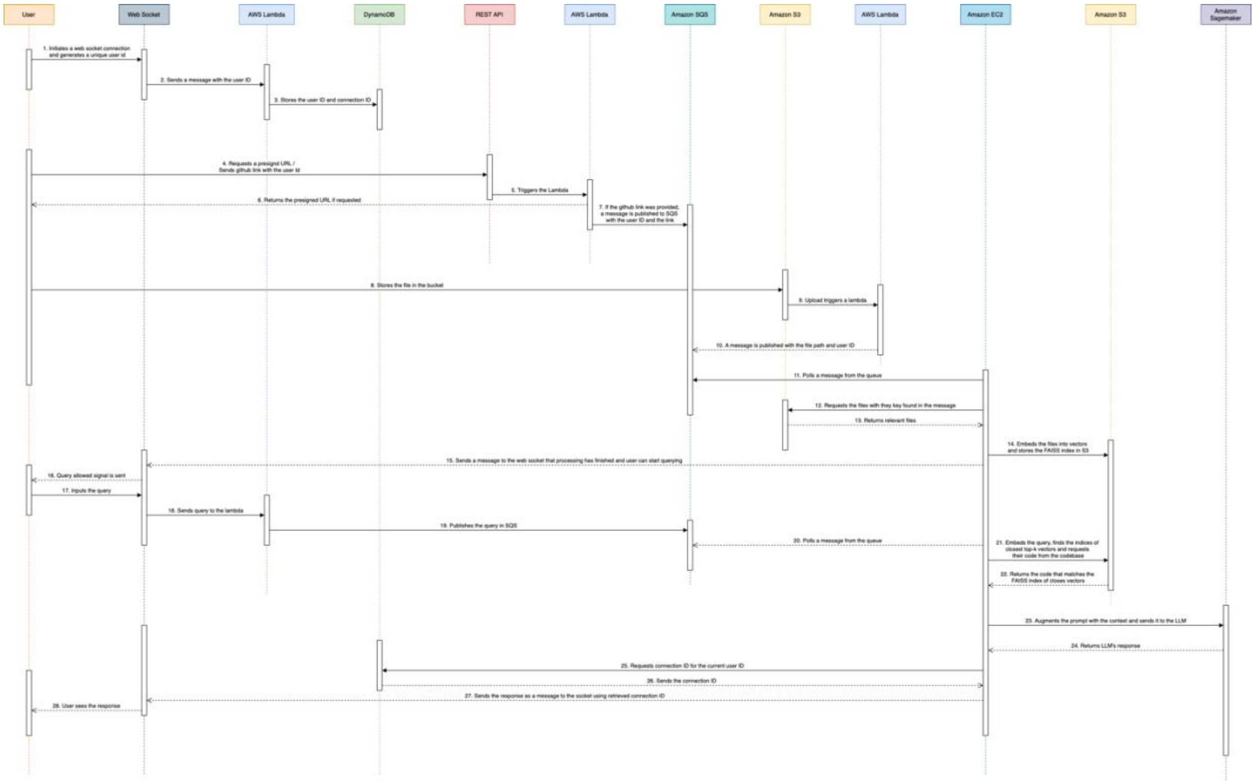

User
Web Socket
AWS Lambda
DynamoDB
REST API
AWS Lambda
Amazon SQS
Amazon S3
AWS Lambda
Amazon EC2
Amazon S3
Amazon Sagemaker
1. Initiates a web socket connection and generates a unique user id
2. Sends a message with the user ID
3. Stores the user ID and connection ID
4. Requests a presignd URL / Sends github link with the user Id
5. Triggers the Lambda
6. Returns the presigned URL if requested
7. If the github link was provided, a message is published to SQS with the user ID and the link
8. Stores the file in the bucket
9. Upload triggers a lambda
10. A message is published with the file path and user ID
11. Polls a message from the queue
12. Requests the files with they key found in the message
13. Returns relevant files
14. Embeds the files into vectors and stores the FAISS index in S3
15. Sends a message to the web socket that processing has finished and user can start querying
16. Query allowed signal is sent
17. Inputs the query
18. Sends query to the lambda
19. Publishes the query in SQS
20. Polls a message from the queue
21. Embeds the query, finds the indices of closest top-k vectors and requests their code from the codebase
22. Returns the code that matches the FAISS index of closes vectors
23. Augments the prompt with the context and sends it to the LLM
24. Returns LLM's response
25. Requests connection ID for the current user ID
26. Sends the connection ID
27. Sends the response as a message to the socket using retrieved connection ID
28. User sees the response

Fig 6.2 UML Data Sequence diagram

7. AWS Services and Tech Stack

Services:

Amazon CloudFront
Amazon S3
API Gateway (REST and Web Socket)
AWS Lambda
Amazon SQS
Amazon DynamoDB
Amazon EC2 (Auto Scaling Groups)
Amazon Sagemaker
Amazon CloudWatch

Programming Languages:

JavaScript for AWS Lambda
Python for EC2 (back-end) code

Additional Tools:

CodeBERT from Hugging Face (for converting text data to vector)
CodeLLama from Meta (to get response for the augmented prompt)

8. Potential Architectural Challenges

- Maintaining long-lived web socket connection, especially when there are multiple concurrent connections, could be tricky because of connection timeouts and scaling issues with concurrent users.

- If the model used to create embeddings gets updated, it can break the semantic alignment. Hence, it's important to keep track of specific version of the model used to create embeddings.

- Users with malicious intent can compromise the system while querying the model. Input sanitization is necessary to avoid this.

- If chunks are too large or too many, program might hit the token limit. It is important to embed the code in smaller chunks.

9. Initial cost estimation for overall implementation

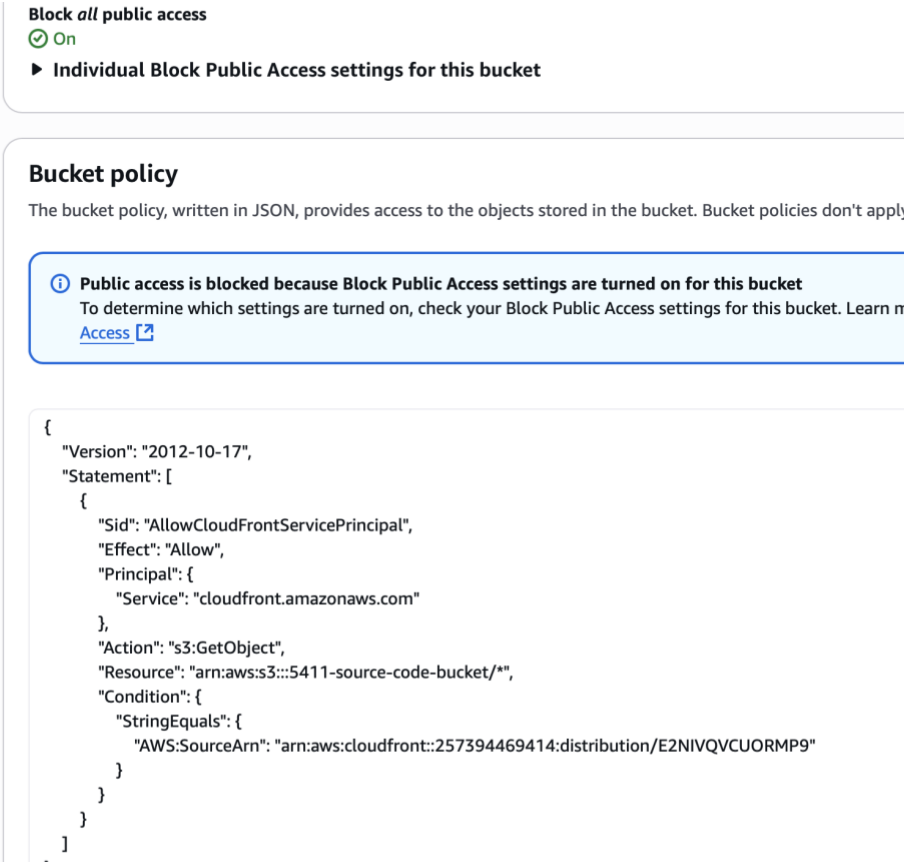
Service	Estimate & Assumptions	Cost*
Amazon S3 (static hosting)	First 50 TB / month + per 1000 requests	$0.023 + $0.0054
CloudFront	First 1 TB monthly transfer + requests	$0
API Gateway (REST + WebSocket)	1M REST calls & 1M WS messages	$3.50 + $1.00
AWS Lambda x 3	Price per milli seconds for 128MB memory functions + 1M requests	$0.0000000021 + $0.20 (x 3)
Amazon SQS	1 M requests + First 10 TB / Month data out	$0.50 + $0.09
Amazon DynamoDB	1M WRU/RRU + per GB per month (First 25 GB is free)	$0.625 + $0.125 + $0.25
Amazon EC2	C6i.2xlarge on demand hourly with 15 GB storage	$0.34
Amazon SageMaker	ml.g5.2xlarge instance for model deployment	$1.212
Amazon S3 (vector store)	First 50 TB / month + per 1000 requests	$0.023 + $0.0054
Data transfer between services	(Mostly free within same AZ, negligible)	$0
Total		**$8.33**

** Costs have been retrieved from AWS's official pricing pages. Free tier for services is ignored.*

10. Explanation

- **Front-end hosted on S3 Bucket**

  The bucket that hosts the front-end code is inaccessible to the outside world. Only the CloudFront distribution can access the bucket contents due to the policy set on the S3 Bucket. The S3 bucket has versioning enabled so that one can always go to the previous version in case of an anomaly.



Block *all* public access

✓ On

▶ **Individual Block Public Access settings for this bucket**

Bucket policy

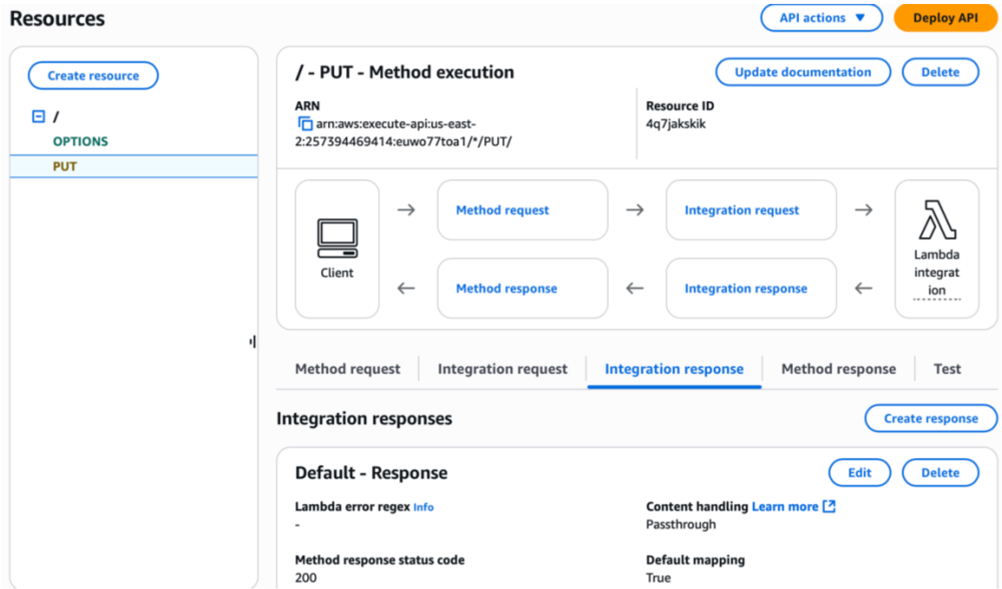
The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't appl

ⓘ **Public access is blocked because Block Public Access settings are turned on for this bucket**
To determine which settings are turned on, check your Block Public Access settings for this bucket. Learn m
Access

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCloudFrontServicePrincipal",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudfront.amazonaws.com"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::5411-source-code-bucket/*",
      "Condition": {
        "StringEquals": {
          "AWS:SourceArn": "arn:aws:cloudfront::257394469414:distribution/E2NIVQVCUORMP9"
        }
      }
    }
  ]
```

- **API Gateway**

  There is a REST API and a Websocket API configured on API Gateway. While it doesn't require authorization to access the API at this point, the URLs for the APIs are stored along with the front-end code in the S3 bucket. This allows the front-end to fetch the API during runtime so the APIs are hidden. Bothe the APIs invoke lambda integrations.



/ - PUT - Method execution

ARN
arn:aws:execute-api:us-east-2:257394469414:euwo77toa1/*/PUT/

Resource ID
4q7jakskik

Integration responses

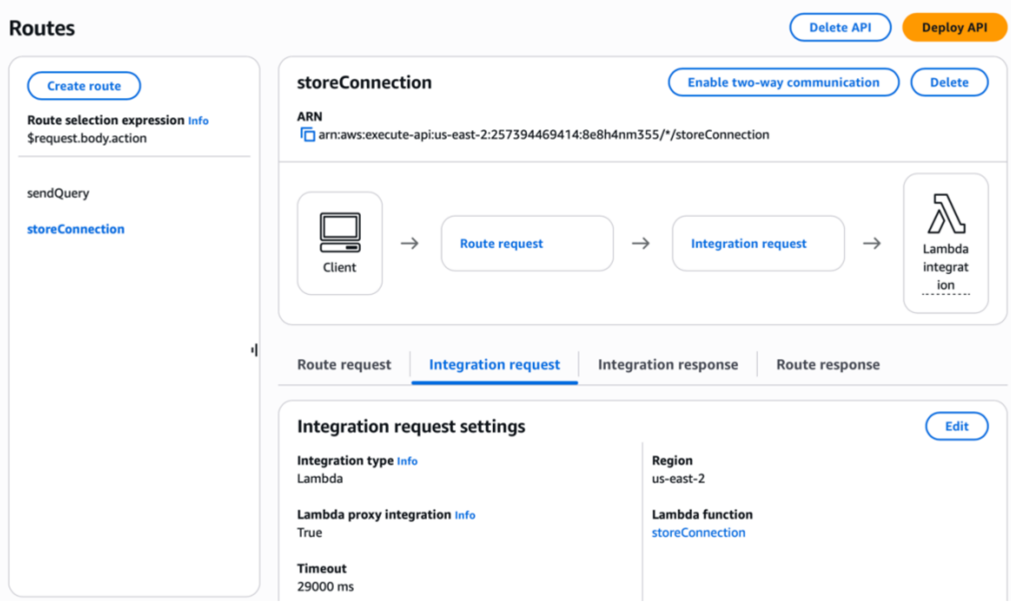
Default - Response

Lambda error regex Info
-

Content handling Learn more
Passthrough

Method response status code
200

Default mapping
True



Route selection expression Info
$request.body.action

storeConnection

ARN
arn:aws:execute-api:us-east-2:257394469414:8e8h4nm355/*/storeConnection

Integration request settings

Integration type Info
Lambda

Region
us-east-2

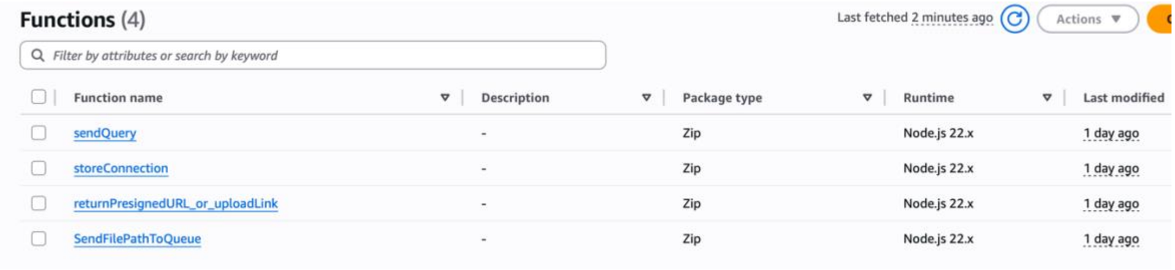
Lambda proxy integration Info
True

Lambda function
storeConnection

Timeout
29000 ms

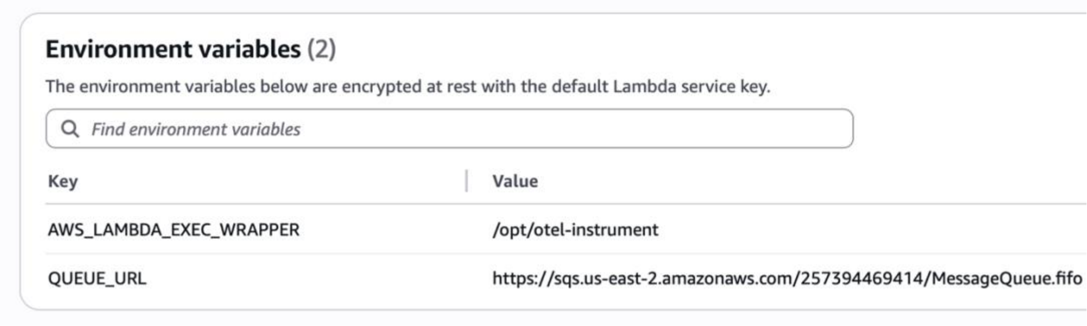
- **Lambda functions**

  There are 4 lambda functions in this architecture. Each has been assigned specific IAM role so that they only have enough access to show logs on CloudWatch, access dynamo DB to store WebSocket connection information, integrate with S3 to generate presigned URLs, and publish messages to the queue (SQS). They have all been assigned environment variables to avoid hardcoding different AWS services URLs. They all run on ARM architecture to save costs.


Last fetched 2 minutes ago
Actions
Filter by attributes or search by keyword

Functions (4)

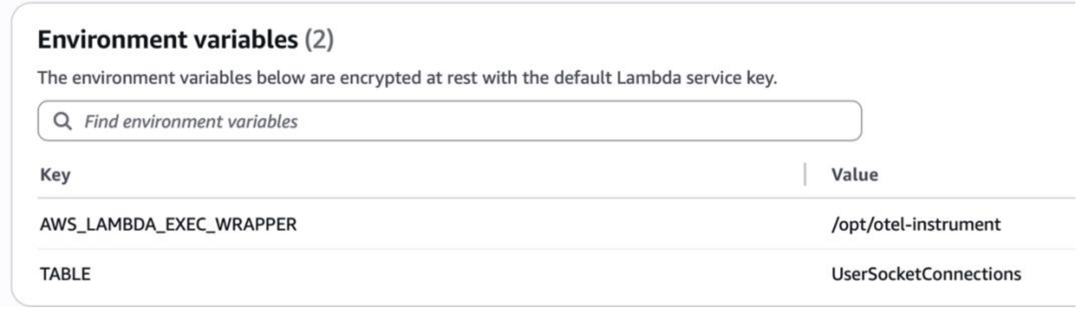
☐	Function name	Description	Package type	Runtime	Last modified
☐	sendQuery	-	Zip	Node.js 22.x	1 day ago
☐	storeConnection	-	Zip	Node.js 22.x	1 day ago
☐	returnPresignedURL_or_uploadLink	-	Zip	Node.js 22.x	1 day ago
☐	SendFilePathToQueue	-	Zip	Node.js 22.x	1 day ago



Environment variables (2)

The environment variables below are encrypted at rest with the default Lambda service key.

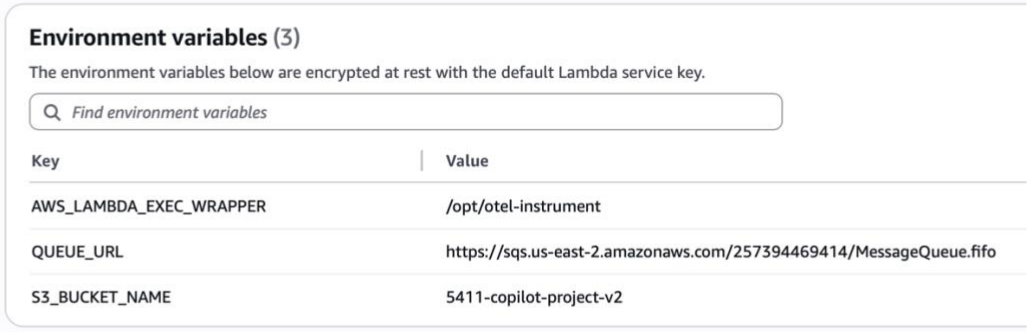
Key	Value
AWS_LAMBDA_EXEC_WRAPPER	/opt/otel-instrument
QUEUE_URL	https://sqs.us-east-2.amazonaws.com/257394469414/MessageQueue.fifo



Environment variables (2)

The environment variables below are encrypted at rest with the default Lambda service key.

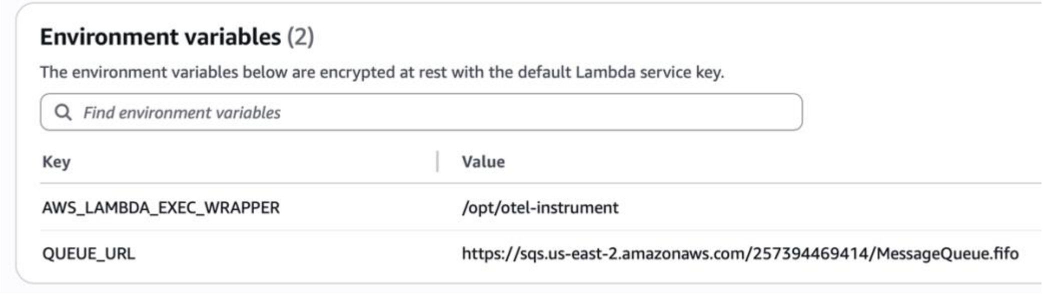
Key	Value
AWS_LAMBDA_EXEC_WRAPPER	/opt/otel-instrument
TABLE	UserSocketConnections



Environment variables (3)

The environment variables below are encrypted at rest with the default Lambda service key.

Key	Value
AWS_LAMBDA_EXEC_WRAPPER	/opt/otel-instrument
QUEUE_URL	https://sqs.us-east-2.amazonaws.com/257394469414/MessageQueue.fifo
S3_BUCKET_NAME	5411-copilot-project-v2



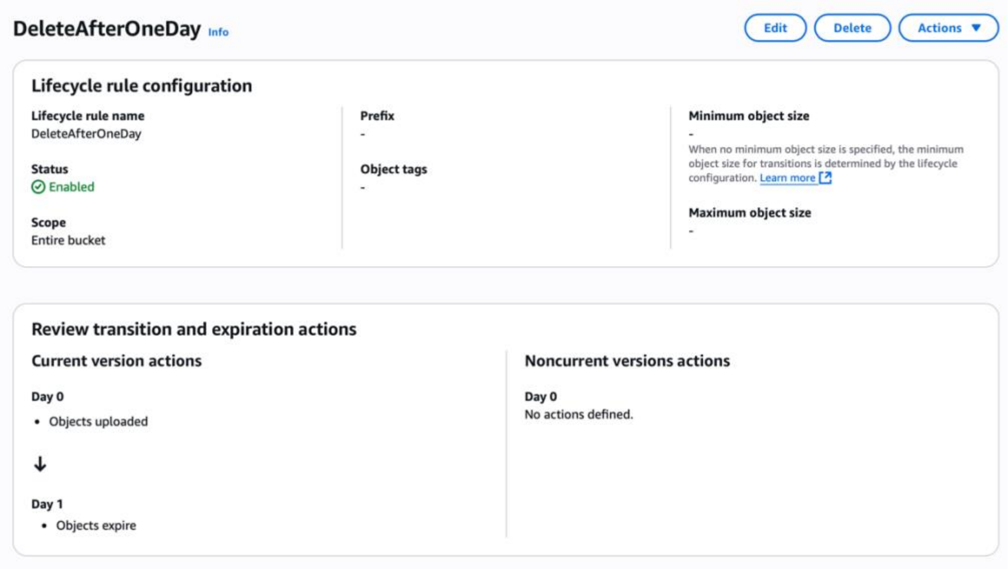
Environment variables (2)

The environment variables below are encrypted at rest with the default Lambda service key.

Key	Value
AWS_LAMBDA_EXEC_WRAPPER	/opt/otel-instrument
QUEUE_URL	https://sqs.us-east-2.amazonaws.com/257394469414/MessageQueue.fifo

- **User upload store bucket**

  This is the bucket where user uploaded files are stored. Each uploaded file is prefixed with a unique userId for efficient retrieval. A lifecycle management rule here deletes all the files after 1 day because user wouldn't need them after a session is over. This bucket is also inaccessible to the outside world and only resources with appropriate role can access this bucket (EC2 instances in this case).



Lifecycle rule configuration

Lifecycle rule name
DeleteAfterOneDay

Status
✓ Enabled

Scope
Entire bucket

Prefix
-

Object tags
-

Minimum object size
-
When no minimum object size is specified, the minimum object size for transitions is determined by the lifecycle configuration. Learn more

Maximum object size
-

Review transition and expiration actions

Current version actions

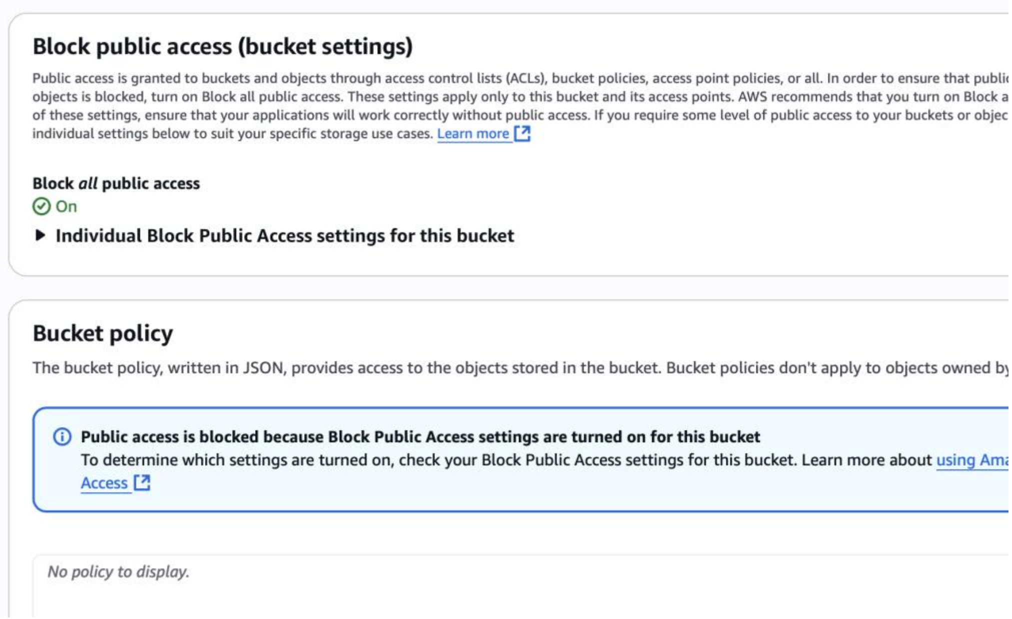
Day 0
- Objects uploaded

↓

Day 1
- Objects expire

Noncurrent versions actions

Day 0
No actions defined.



Block public access (bucket settings)

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that publi objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block a of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objec individual settings below to suit your specific storage use cases. Learn more

Block *all* public access
✓ On

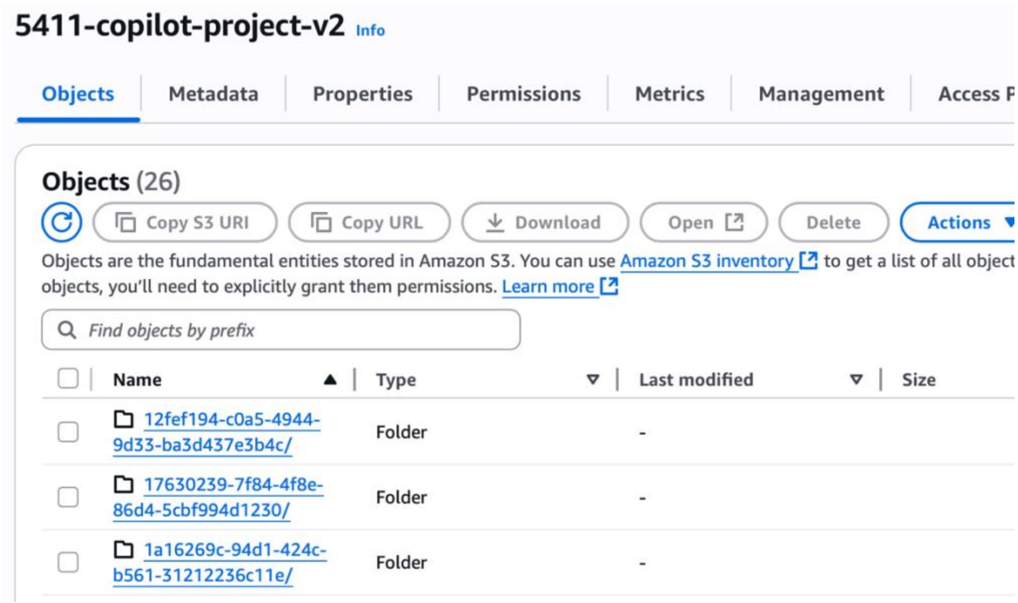
▶ Individual Block Public Access settings for this bucket

Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned b

ⓘ **Public access is blocked because Block Public Access settings are turned on for this bucket**
To determine which settings are turned on, check your Block Public Access settings for this bucket. Learn more about using Ama Access

No policy to display.



5411-copilot-project-v2

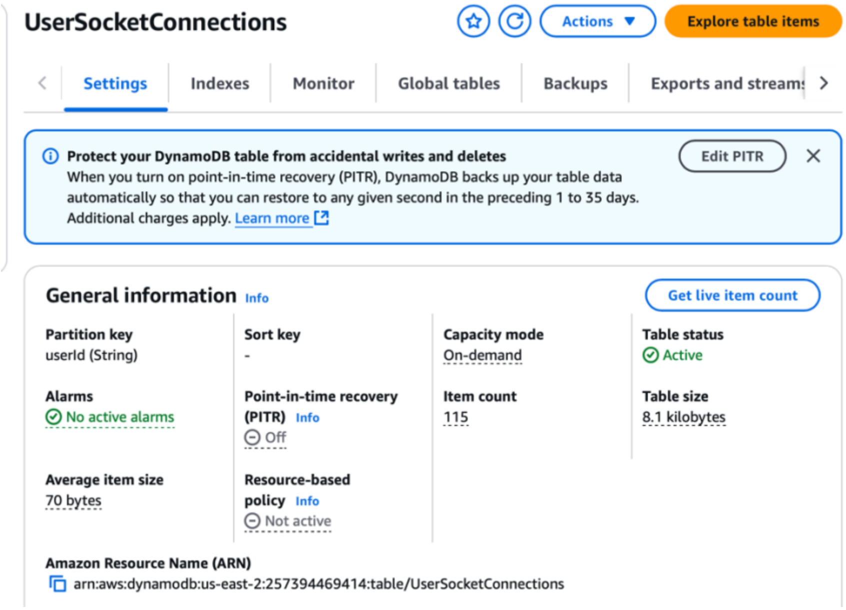
Objects (26)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all object
objects, you'll need to explicitly grant them permissions. Learn more

□	Name ▲	Type	Last modified	Size
□	12fef194-c0a5-4944-9d33-ba3d437e3b4c/	Folder	-	
□	17630239-7f84-4f8e-86d4-5cbf994d1230/	Folder	-	
□	1a16269c-94d1-424c-b561-31212236c11e/	Folder	-	

- **Dynamo DB**

UserSocketConnection table stores the user to connection mapping. There is no resource based policy attached to it.



UserSocketConnections

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 1 to 35 days. Additional charges apply. Learn more

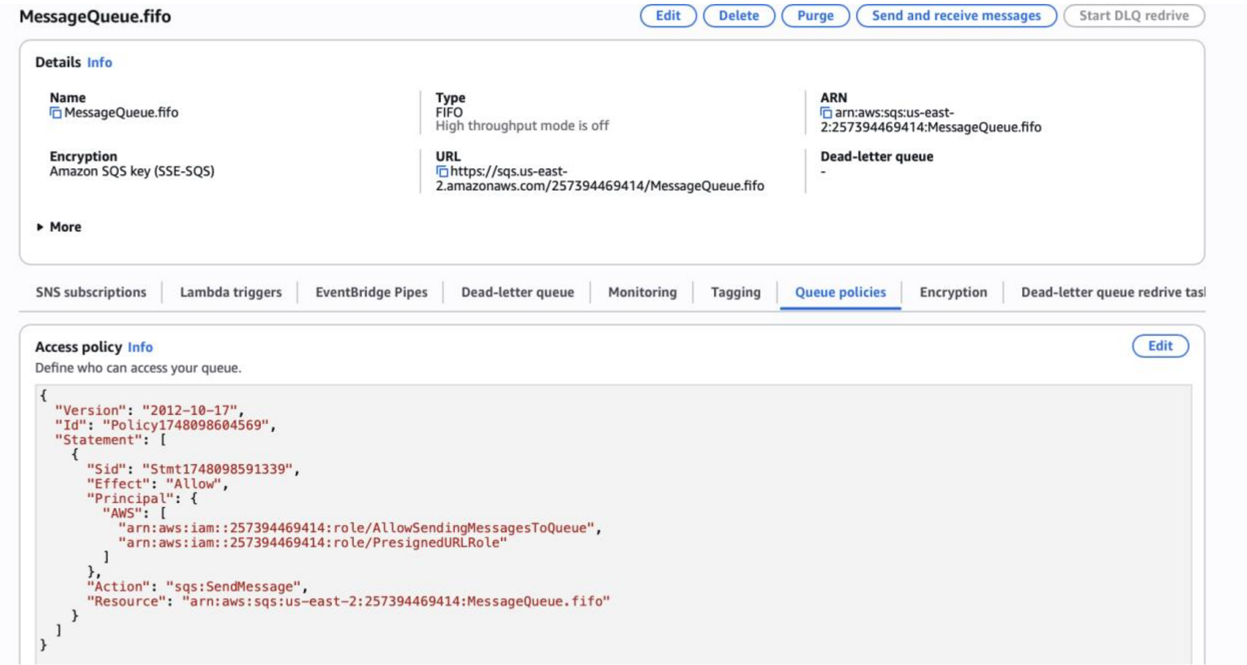
General information Info

Partition key: userId (String)
Sort key: -
Capacity mode: On-demand
Table status: Active
Alarms: No active alarms
Point-in-time recovery (PITR) Info: Off
Item count: 115
Table size: 8.1 kilobytes
Average item size: 70 bytes
Resource-based policy Info: Not active

Amazon Resource Name (ARN)
arn:aws:dynamodb:us-east-2:257394469414:table/UserSocketConnections

- **SQS**

This queue stores messages which has URL of files in S3 or GitHub link, and user query. It has a resource-based policy attached to it so that only resources (lambda functions) with appropriate role can access it.



Details Info

Name
MessageQueue.fifo

Type
FIFO
High throughput mode is off

ARN
arn:aws:sqs:us-east-2:257394469414:MessageQueue.fifo

Encryption
Amazon SQS key (SSE-SQS)

URL
https://sqs.us-east-2.amazonaws.com/257394469414/MessageQueue.fifo

Dead-letter queue
-

▸ More

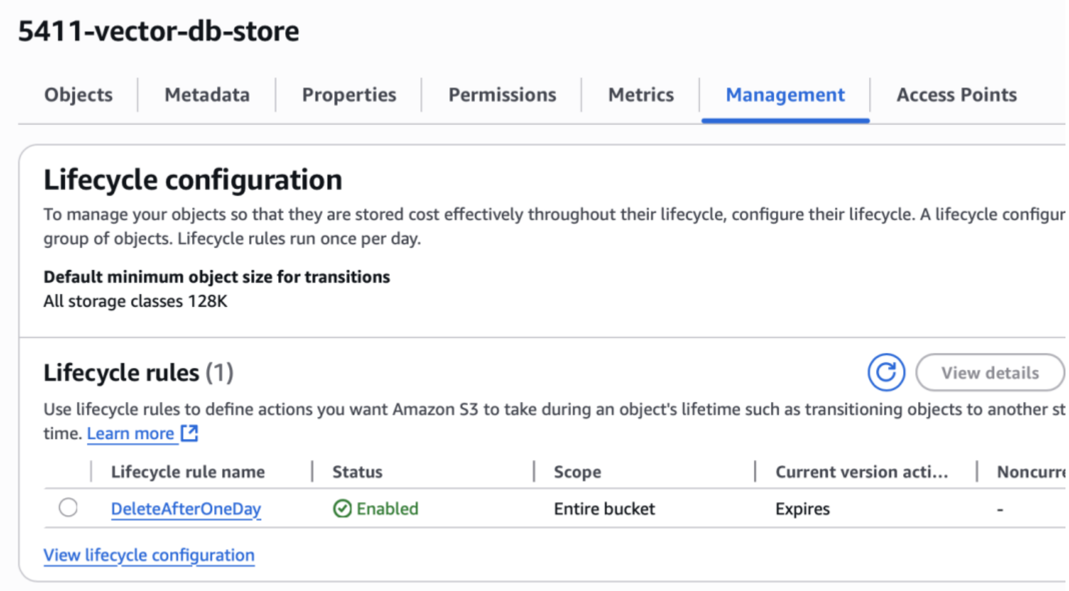
Access policy Info

Define who can access your queue.

```
{
  "Version": "2012-10-17",
  "Id": "Policy1748098604569",
  "Statement": [
    {
      "Sid": "Stmt1748098591339",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::257394469414:role/AllowSendingMessagesToQueue",
          "arn:aws:iam::257394469414:role/PresignedURLRole"
        ]
      },
      "Action": "sqs:SendMessage",
      "Resource": "arn:aws:sqs:us-east-2:257394469414:MessageQueue.fifo"
    }
  ]
}
```

- **S3 vector store bucket**

This bucket stores the vectors computed for files uploaded by users. It also has a lifecycle rule that deletes all the files after one day, and it's public access is also blocked by default.



Lifecycle configuration

To manage your objects so that they are stored cost effectively throughout their lifecycle, configure their lifecycle. A lifecycle configur group of objects. Lifecycle rules run once per day.

Default minimum object size for transitions
All storage classes 128K

Lifecycle rules (1)

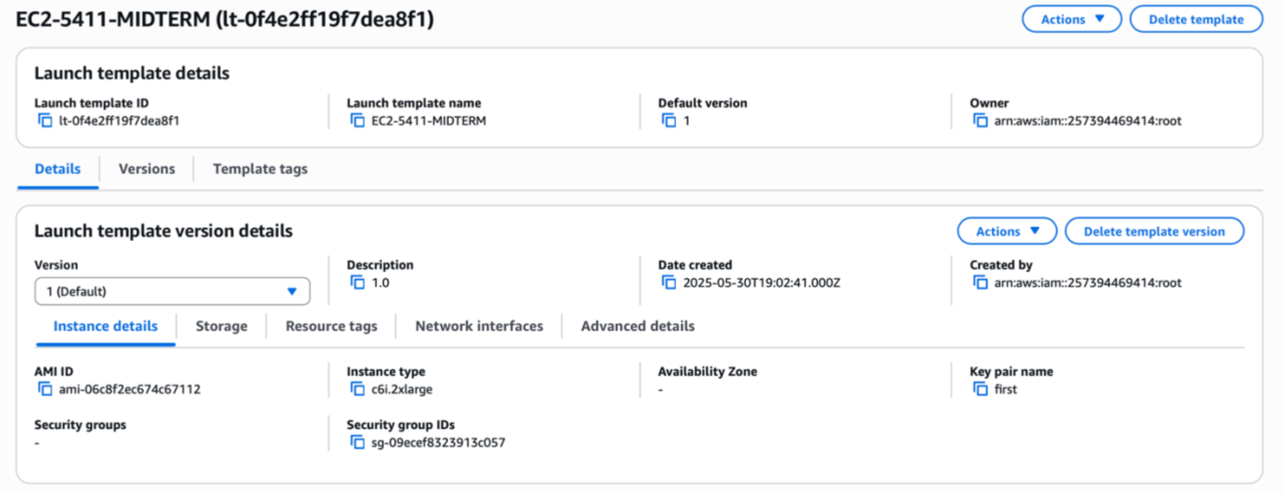
Use lifecycle rules to define actions you want Amazon S3 to take during an object's lifetime such as transitioning objects to another st time. Learn more

	Lifecycle rule name	Status	Scope	Current version acti…	Noncurr
○	DeleteAfterOneDay	✓ Enabled	Entire bucket	Expires	-

- **Amazon EC2**

The Auto Scaling Group launches instances in a private subnet. Each instance runs a user data script that retrieves the back-end code from an S3 bucket, installs the necessary dependencies, and starts the Python application. These EC2 instances do not have public IPv4 access. The Auto

Scaling Group monitors the SQS queue, and if the number of messages exceeds five, it provisions additional instances. The attached security group restricts traffic to outbound-only access. C6i instances are used to support multiple worker threads and handle transformer models efficiently. However, instance size may be reduced in the future based on CPU and memory usage analysis.



EC2-5411-MIDTERM (lt-0f4e2ff19f7dea8f1)

Launch template details

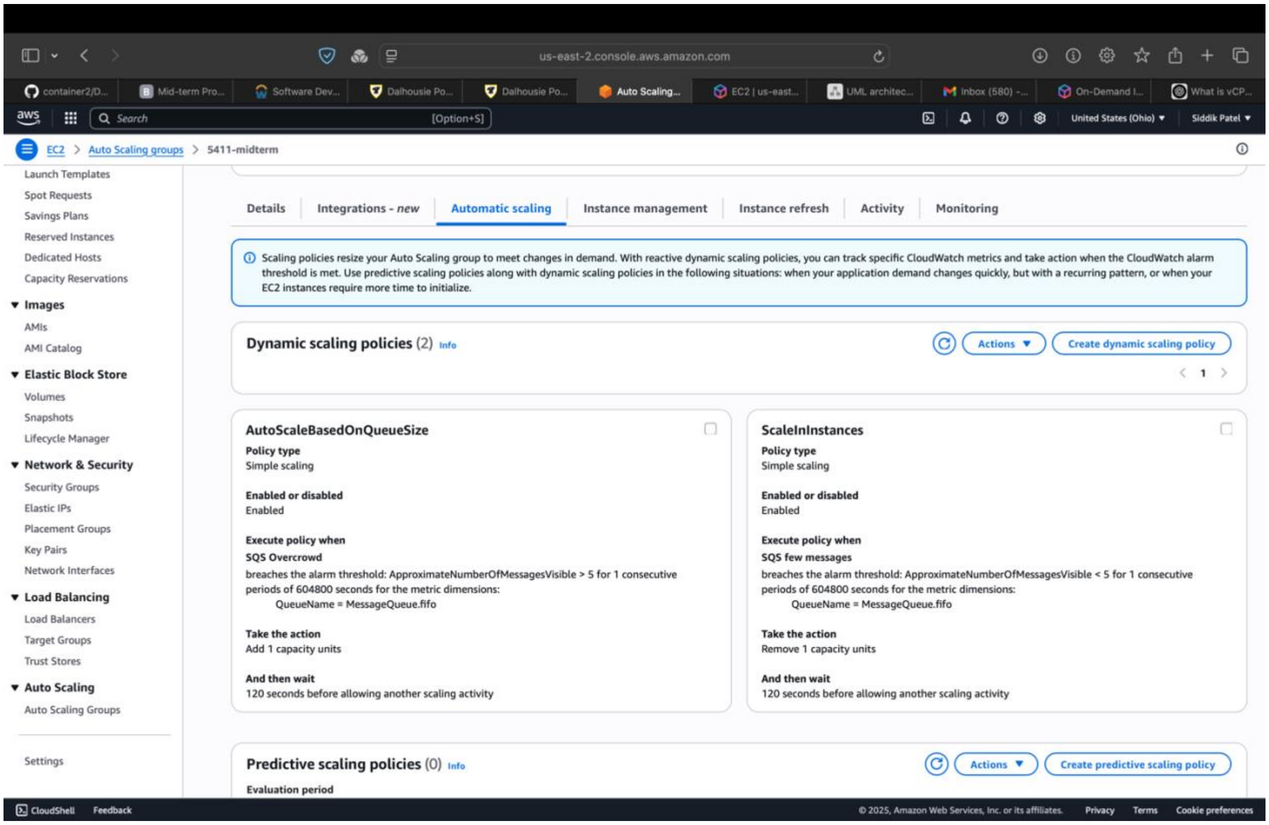
Launch template ID	Launch template name	Default version	Owner
lt-0f4e2ff19f7dea8f1	EC2-5411-MIDTERM	1	arn:aws:iam::257394469414:root

Launch template version details

Version	Description	Date created	Created by
1 (Default)	1.0	2025-05-30T19:02:41.000Z	arn:aws:iam::257394469414:root

AMI ID	Instance type	Availability Zone	Key pair name
ami-06c8f2ec674c67112	c6i.2xlarge	-	first

Security groups	Security group IDs
-	sg-09ecef8323913c057



Scaling policies resize your Auto Scaling group to meet changes in demand. With reactive dynamic scaling policies, you can track specific CloudWatch metrics and take action when the CloudWatch alarm threshold is met. Use predictive scaling policies along with dynamic scaling policies in the following situations: when your application demand changes quickly, but with a recurring pattern, or when your EC2 instances require more time to initialize.

Dynamic scaling policies (2) Info

AutoScaleBasedOnQueueSize

Policy type
Simple scaling

Enabled or disabled
Enabled

Execute policy when
SQS Overcrowd
breaches the alarm threshold: ApproximateNumberOfMessagesVisible > 5 for 1 consecutive periods of 604800 seconds for the metric dimensions:
QueueName = MessageQueue.fifo

Take the action
Add 1 capacity units

And then wait
120 seconds before allowing another scaling activity

ScaleInInstances

Policy type
Simple scaling

Enabled or disabled
Enabled

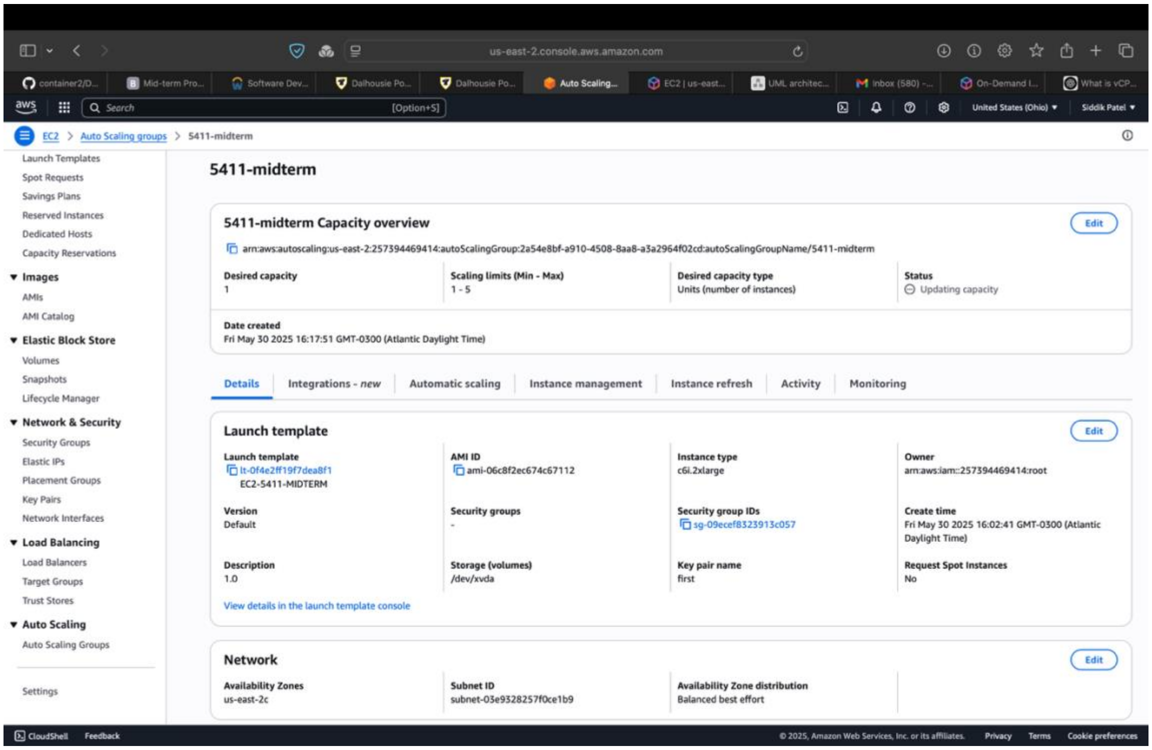
Execute policy when
SQS few messages
breaches the alarm threshold: ApproximateNumberOfMessagesVisible < 5 for 1 consecutive periods of 604800 seconds for the metric dimensions:
QueueName = MessageQueue.fifo

Take the action
Remove 1 capacity units

And then wait
120 seconds before allowing another scaling activity

Predictive scaling policies (0) Info

Evaluation period



5411-midterm

5411-midterm Capacity overview

arn:aws:autoscaling:us-east-2:257394469414:autoScalingGroup:2a54e8bf-a910-4508-8aa8-a3a2964f02cd:autoScalingGroupName/5411-midterm

Desired capacity	Scaling limits (Min - Max)	Desired capacity type	Status
1	1 - 5	Units (number of instances)	Updating capacity

Date created
Fri May 30 2025 16:17:51 GMT-0300 (Atlantic Daylight Time)

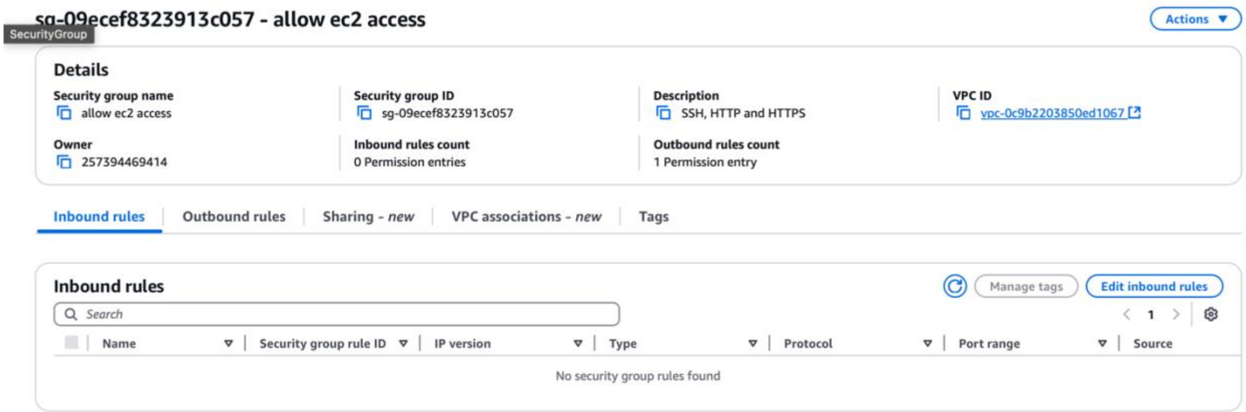
Launch template

Launch template	AMI ID	Instance type	Owner
lt-0f4e2ff19f7dea8f1 EC2-5411-MIDTERM	ami-06c8f2ec674c67112	c6i.2xlarge	arn:aws:iam::257394469414:root
Version	**Security groups**	**Security group IDs**	**Create time**
Default	-	sg-09ecef8323913c057	Fri May 30 2025 16:02:41 GMT-0300 (Atlantic Daylight Time)
Description	**Storage (volumes)**	**Key pair name**	**Request Spot Instances**
1.0	/dev/xvda	first	No

View details in the launch template console

Network

Availability Zones	Subnet ID	Availability Zone distribution
us-east-2c	subnet-03e9328257f0ce1b9	Balanced best effort



SecurityGroup
Actions ▾
Inbound rules | Outbound rules | Sharing - *new* | VPC associations - *new* | Tags
Manage tags
Edit inbound rules
Search
1

sg-09ecef8323913c057 - allow ec2 access

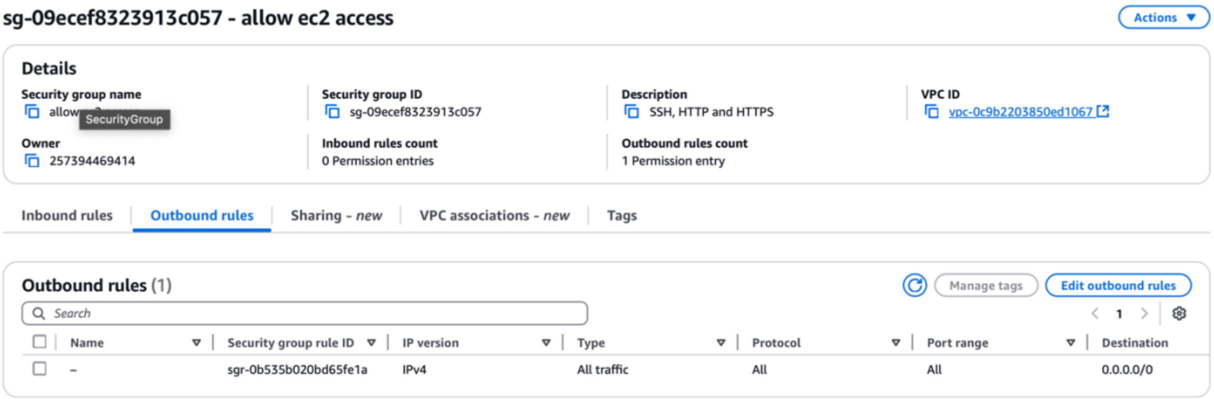
Details

Security group name	Security group ID	Description	VPC ID
allow ec2 access	sg-09ecef8323913c057	SSH, HTTP and HTTPS	vpc-0c9b2203850ed1067
Owner	**Inbound rules count**	**Outbound rules count**	
257394469414	0 Permission entries	1 Permission entry	

Inbound rules

□	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source

No security group rules found



SecurityGroup
Actions ▾
Inbound rules | Outbound rules | Sharing - *new* | VPC associations - *new* | Tags
Manage tags
Edit outbound rules
Search
1

sg-09ecef8323913c057 - allow ec2 access

Details

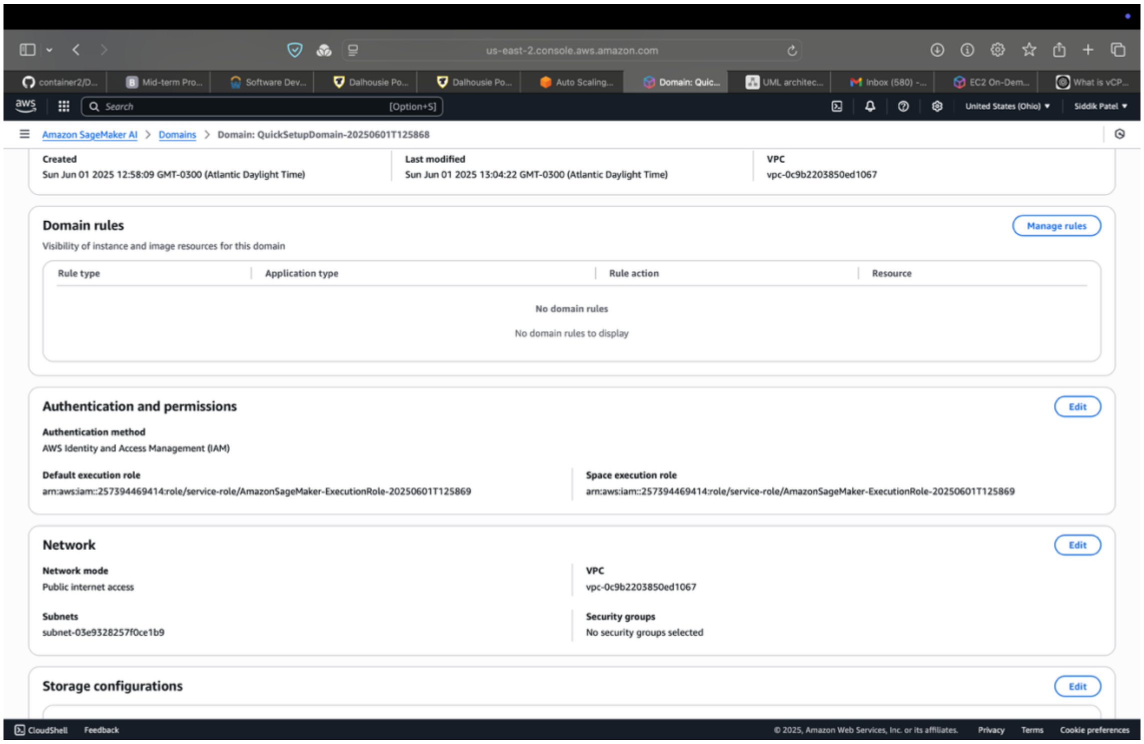
Security group name	Security group ID	Description	VPC ID
allow	sg-09ecef8323913c057	SSH, HTTP and HTTPS	vpc-0c9b2203850ed1067
Owner	**Inbound rules count**	**Outbound rules count**	
257394469414	0 Permission entries	1 Permission entry	

Outbound rules (1)

□	Name	Security group rule ID	IP version	Type	Protocol	Port range	Destination
□	–	sgr-0b535b020bd65fe1a	IPv4	All traffic	All	All	0.0.0.0/0

- **Amazon SageMaker AI**

The SageMaker domain's network is also deployed in the private subnet where the security rules doesn't allow inbound access from the network. The CodeLlama model is deployed on a default endpoint. The endpoint is hosted on ml.g5.2xlarge instance because of the CodeLlama model's compatibility.



Created
Sun Jun 01 2025 12:58:09 GMT-0300 (Atlantic Daylight Time)

Last modified
Sun Jun 01 2025 13:04:22 GMT-0300 (Atlantic Daylight Time)

VPC
vpc-0c9b2203850ed1067

Domain rules

Visibility of instance and image resources for this domain

Rule type	Application type	Rule action	Resource

No domain rules

No domain rules to display

Authentication and permissions

Authentication method
AWS Identity and Access Management (IAM)

Default execution role
arn:aws:iam::257394469414:role/service-role/AmazonSageMaker-ExecutionRole-20250601T125869

Space execution role
arn:aws:iam::257394469414:role/service-role/AmazonSageMaker-ExecutionRole-20250601T125869

Network

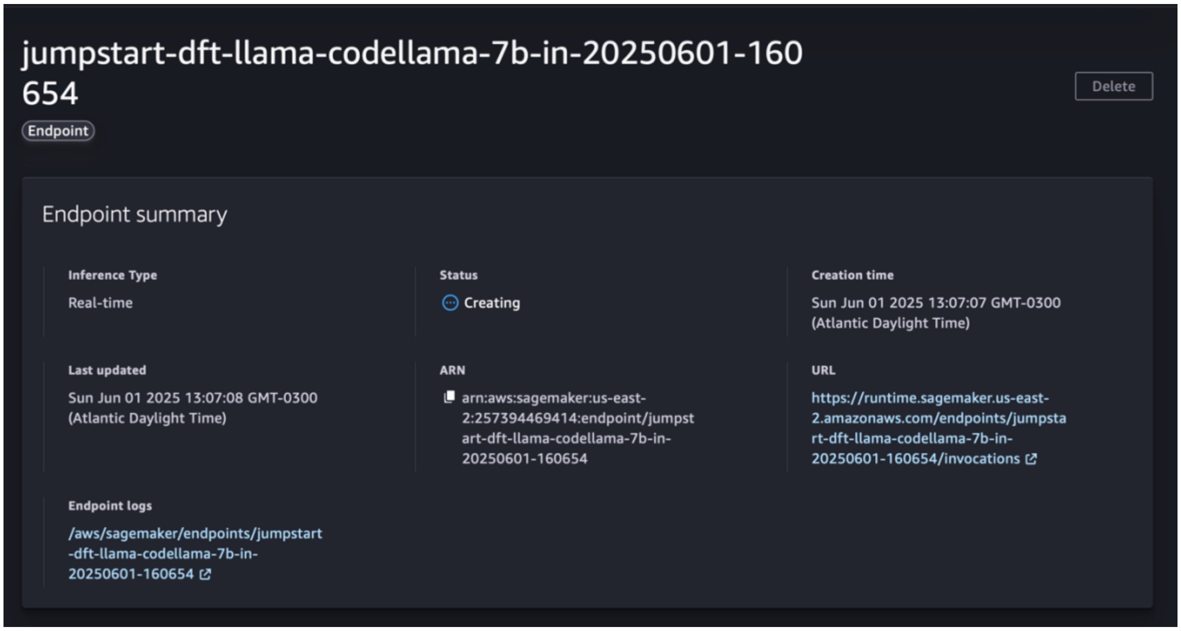
Network mode
Public internet access

VPC
vpc-0c9b2203850ed1067

Subnets
subnet-03e9328257f0ce1b9

Security groups
No security groups selected

Storage configurations



jumpstart-dft-llama-codellama-7b-in-20250601-160654

Endpoint summary

Inference Type
Real-time

Status
Creating

Creation time
Sun Jun 01 2025 13:07:07 GMT-0300 (Atlantic Daylight Time)

Last updated
Sun Jun 01 2025 13:07:08 GMT-0300 (Atlantic Daylight Time)

ARN
arn:aws:sagemaker:us-east-2:257394469414:endpoint/jumpstart-dft-llama-codellama-7b-in-20250601-160654

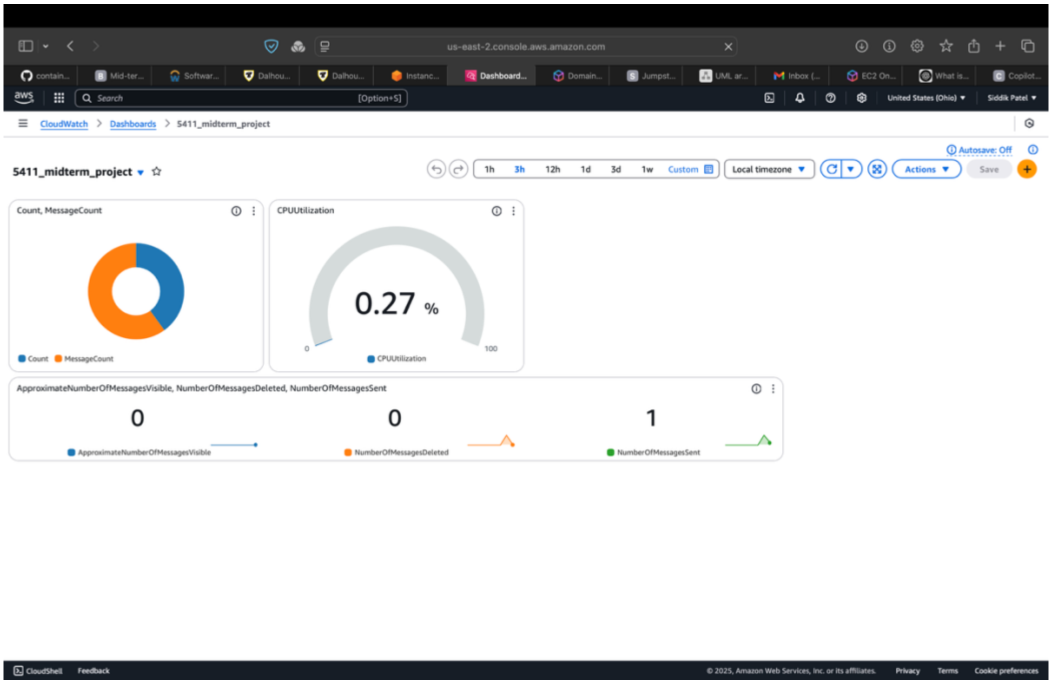
URL
https://runtime.sagemaker.us-east-2.amazonaws.com/endpoints/jumpstart-dft-llama-codellama-7b-in-20250601-160654/invocations

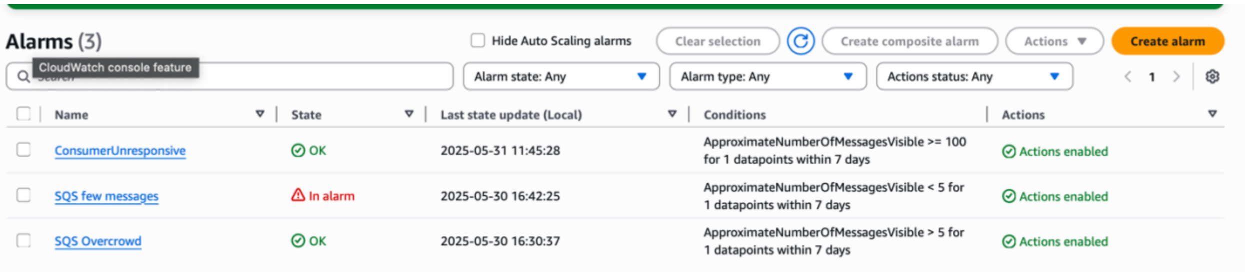
Endpoint logs
/aws/sagemaker/endpoints/jumpstart-dft-llama-codellama-7b-in-20250601-160654

- **Amazon CloudWatch Logs**

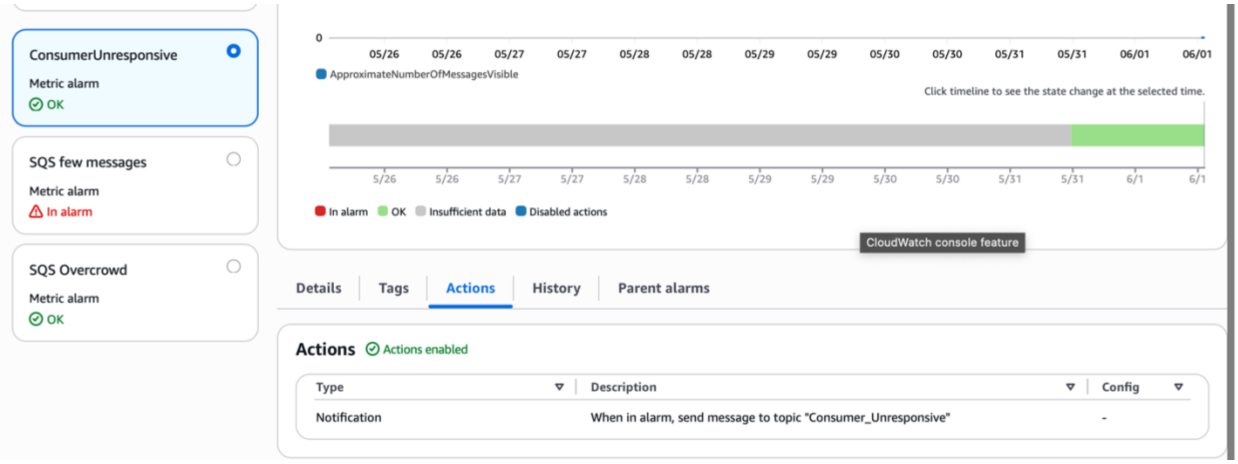
All Lambda functions and the API Gateway generate logs for each request. Alarms configured for SQS trigger EC2 auto scaling when thresholds are met. The monitoring dashboard displays key metrics, including the number of messages in the queue, EC2 instance CPU utilization, and the number of requests received by the API Gateway.

If the back end becomes inactive for any reason and messages begin to accumulate, an SNS topic sends email notifications to alert me. Application traces provide visibility into the availability of Lambda functions.


us-east-2.console.aws.amazon.com
contain... Mid-ter... Softwar... Dalhou... Dalhou... Instanc... Dashboard... Domain... Jumpst... UML ar... Inbox (... EC2 On... What is... Copilot...
aws
Search [Option+S]
United States (Ohio)
Siddik Patel
CloudWatch > Dashboards > 5411_midterm_project
Autosave: Off
5411_midterm_project
1h 3h 12h 1d 3d 1w Custom
Local timezone
Actions
Save
Count, MessageCount
Count
MessageCount
CPUUtilization
0.27 %
0
100
CPUUtilization
ApproximateNumberOfMessagesVisible, NumberOfMessagesDeleted, NumberOfMessagesSent
0
ApproximateNumberOfMessagesVisible
0
NumberOfMessagesDeleted
1
NumberOfMessagesSent
CloudShell Feedback
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



☐	Name	State	Last state update (Local)	Conditions	Actions
☐	ConsumerUnresponsive	✓ OK	2025-05-31 11:45:28	ApproximateNumberOfMessagesVisible >= 100 for 1 datapoints within 7 days	✓ Actions enabled
☐	SQS few messages	⚠ In alarm	2025-05-30 16:42:25	ApproximateNumberOfMessagesVisible < 5 for 1 datapoints within 7 days	✓ Actions enabled
☐	SQS Overcrowd	✓ OK	2025-05-30 16:30:37	ApproximateNumberOfMessagesVisible > 5 for 1 datapoints within 7 days	✓ Actions enabled



ConsumerUnresponsive ●
Metric alarm
✓ OK

SQS few messages ○
Metric alarm
⚠ In alarm

SQS Overcrowd ○
Metric alarm
✓ OK

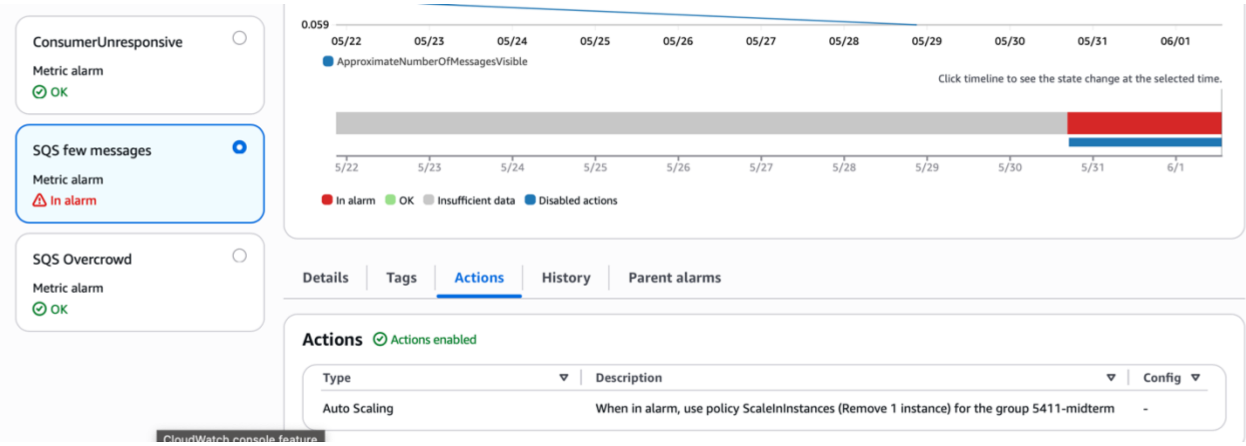
Click timeline to see the state change at the selected time.

🟥 In alarm 🟩 OK ▢ Insufficient data 🟦 Disabled actions

Details | Tags | **Actions** | History | Parent alarms

Actions ✓ Actions enabled

Type	Description	Config
Notification	When in alarm, send message to topic "Consumer_Unresponsive"	-



ConsumerUnresponsive ○
Metric alarm
✓ OK

SQS few messages ●
Metric alarm
⚠ In alarm

SQS Overcrowd ○
Metric alarm
✓ OK

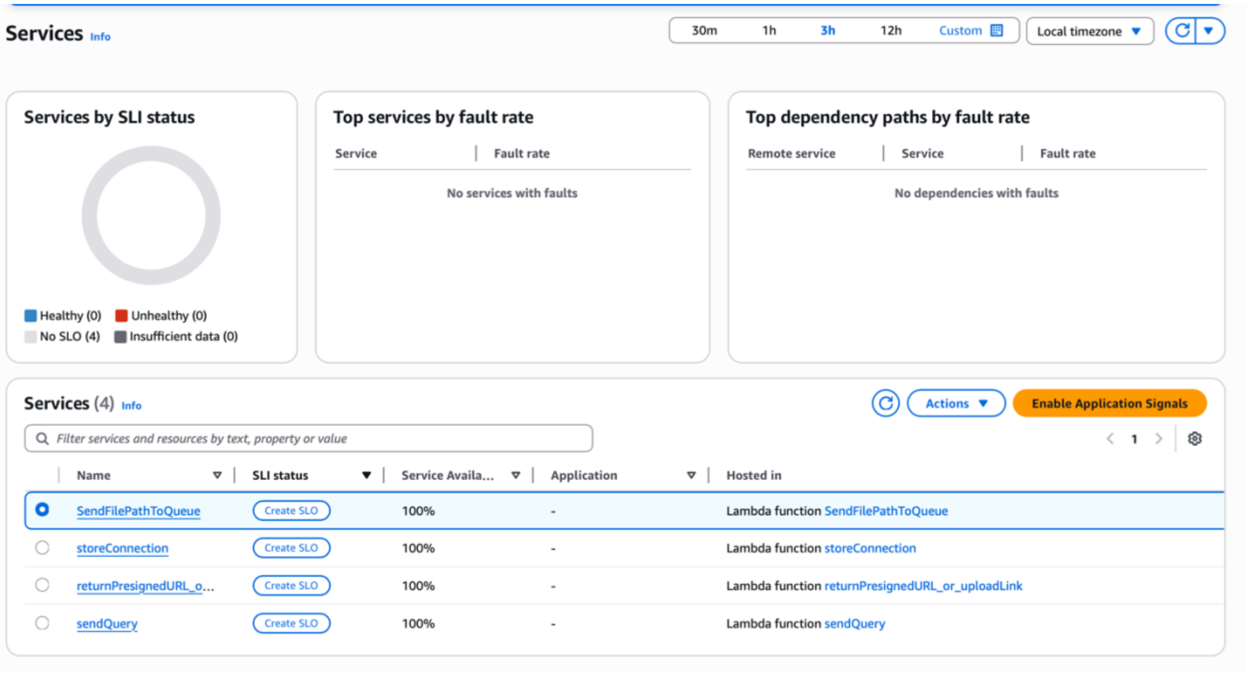
Click timeline to see the state change at the selected time.

🟥 In alarm 🟩 OK ▢ Insufficient data 🟦 Disabled actions

Details | Tags | **Actions** | History | Parent alarms

Actions ✓ Actions enabled

Type	Description	Config
Auto Scaling	When in alarm, use policy ScaleInInstances (Remove 1 instance) for the group 5411-midterm	-



Services Info

Services by SLI status

🟦 Healthy (0) 🟥 Unhealthy (0)
▢ No SLO (4) ■ Insufficient data (0)

Top services by fault rate

Service	Fault rate

No services with faults

Top dependency paths by fault rate

Remote service	Service	Fault rate

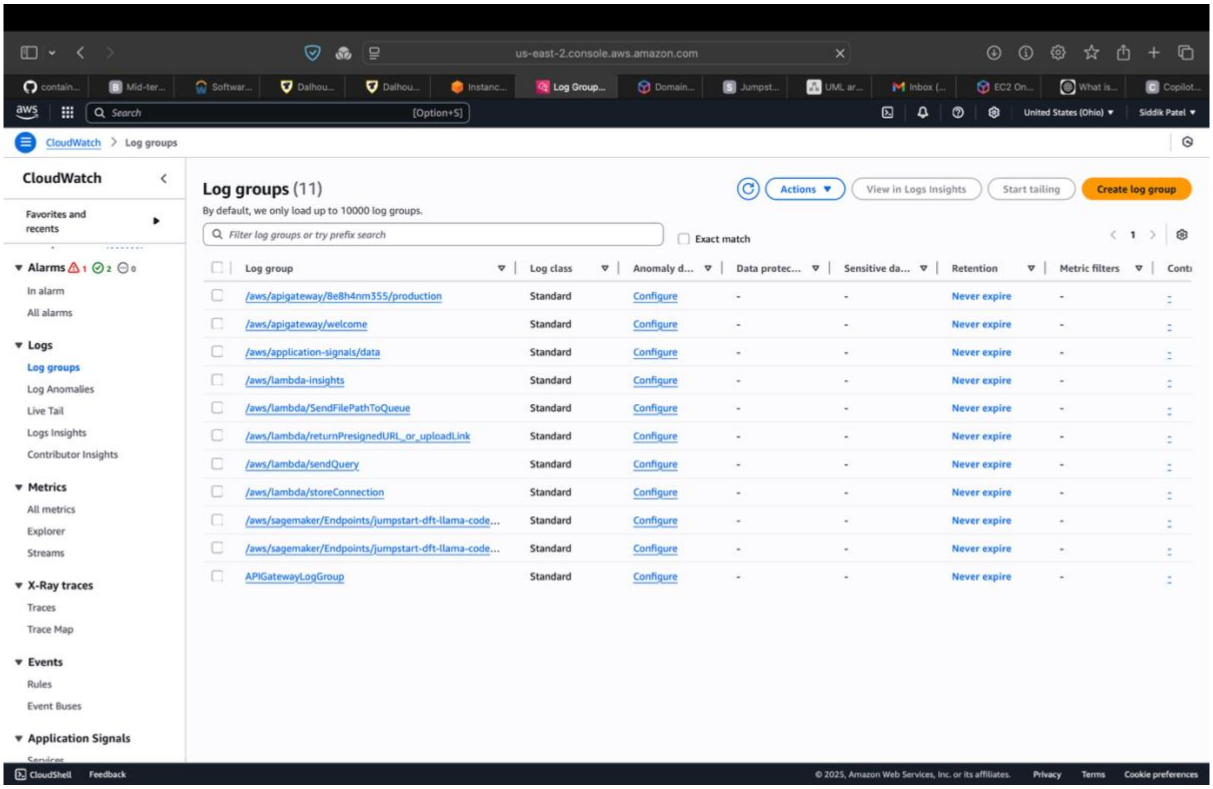
No dependencies with faults

Services (4) Info

Filter services and resources by text, property or value

	Name	SLI status	Service Availa...	Application	Hosted in
●	SendFilePathToQueue	Create SLO	100%	-	Lambda function SendFilePathToQueue
○	storeConnection	Create SLO	100%	-	Lambda function storeConnection
○	returnPresignedURL_o...	Create SLO	100%	-	Lambda function returnPresignedURL_or_uploadLink
○	sendQuery	Create SLO	100%	-	Lambda function sendQuery

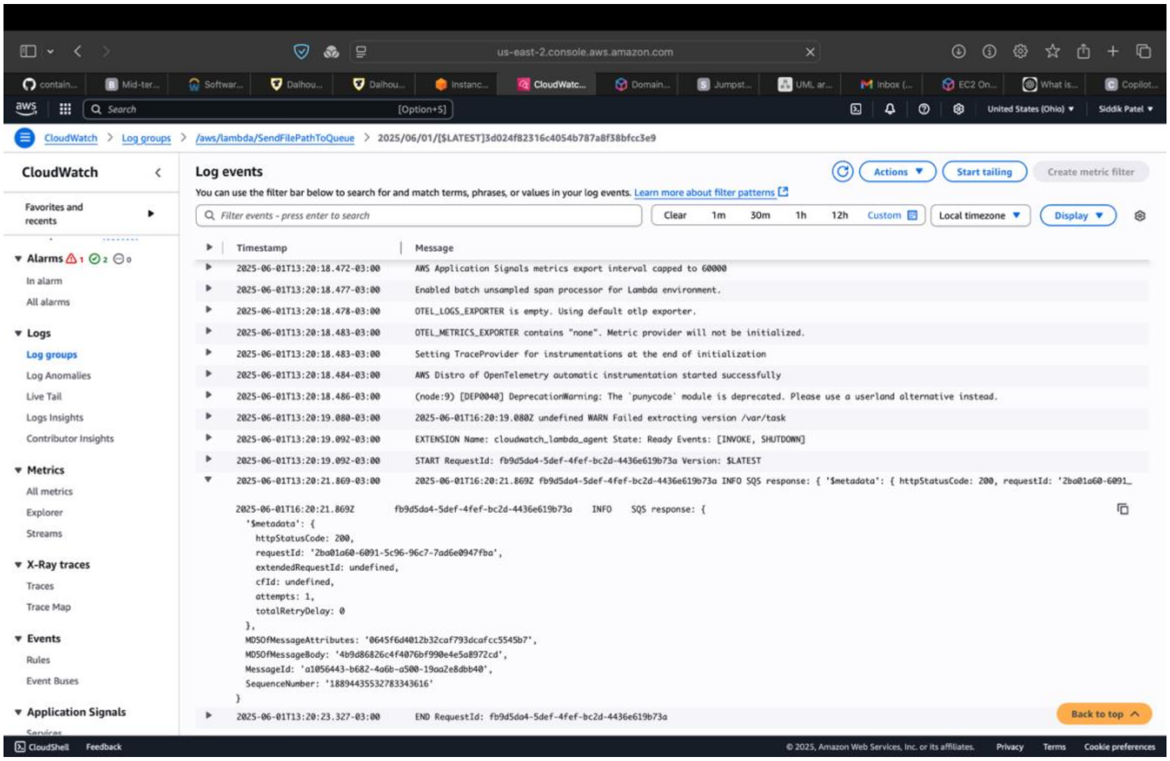
These are the log groups that contain logs from lambda functions, applications signals and API gateways.



Log groups (11)

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

Log group	Log class	Anomaly d...	Data protec...	Sensitive da...	Retention	Metric filters	Conti
/aws/apigateway/8e8h4nm355/production	Standard	Configure	-	-	Never expire	-	-
/aws/apigateway/welcome	Standard	Configure	-	-	Never expire	-	-
/aws/application-signals/data	Standard	Configure	-	-	Never expire	-	-
/aws/lambda-insights	Standard	Configure	-	-	Never expire	-	-
/aws/lambda/SendFilePathToQueue	Standard	Configure	-	-	Never expire	-	-
/aws/lambda/returnPresignedURL_or_uploadLink	Standard	Configure	-	-	Never expire	-	-
/aws/lambda/sendQuery	Standard	Configure	-	-	Never expire	-	-
/aws/lambda/storeConnection	Standard	Configure	-	-	Never expire	-	-
/aws/sagemaker/Endpoints/jumpstart-dft-llama-code...	Standard	Configure	-	-	Never expire	-	-
/aws/sagemaker/Endpoints/jumpstart-dft-llama-code...	Standard	Configure	-	-	Never expire	-	-
APIGatewayLogGroup	Standard	Configure	-	-	Never expire	-	-



CloudWatch > Log groups > /aws/lambda/SendFilePathToQueue > 2025/06/01/[$LATEST]3d024f82316c4054b787a8f38bfcc3e9
Actions
Start tailing
Create metric filter
Clear
1m
30m
1h
12h
Custom
Local timezone
Display
Back to top

Log events

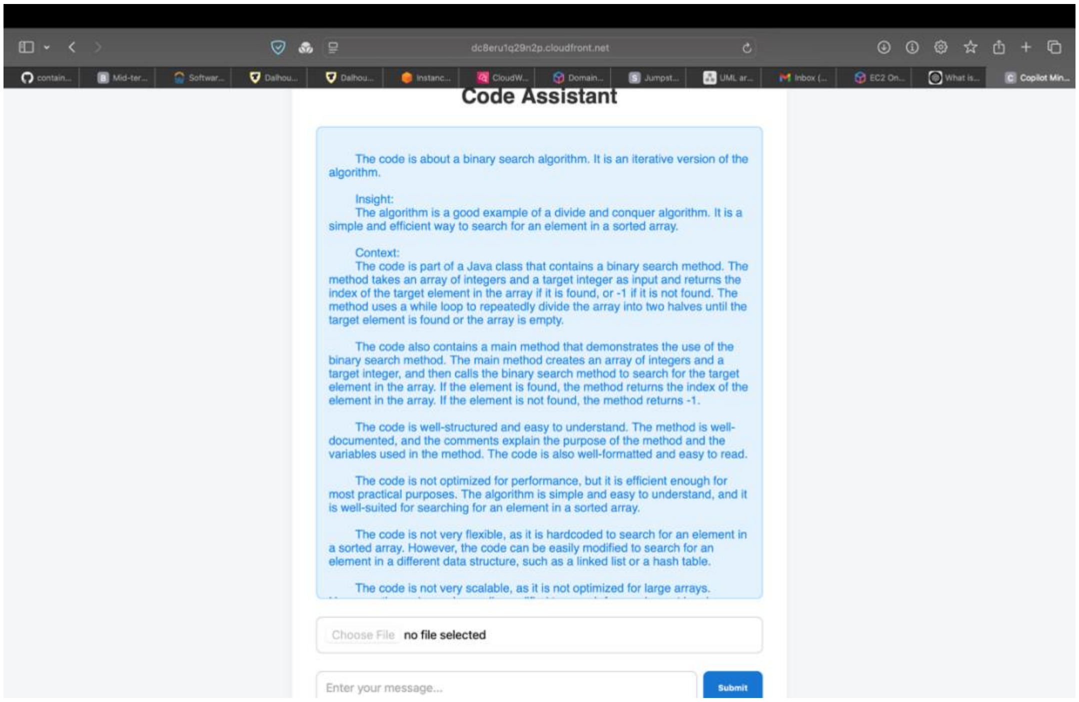
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

```
Timestamp                      Message
2025-06-01T13:20:18.472-03:00  AWS Application Signals metrics export interval capped to 60000
2025-06-01T13:20:18.477-03:00  Enabled batch unsampled span processor for Lambda environment.
2025-06-01T13:20:18.478-03:00  OTEL_LOGS_EXPORTER is empty. Using default otlp exporter.
2025-06-01T13:20:18.483-03:00  OTEL_METRICS_EXPORTER contains "none". Metric provider will not be initialized.
2025-06-01T13:20:18.483-03:00  Setting TraceProvider for instrumentations at the end of initialization
2025-06-01T13:20:18.484-03:00  AWS Distro of OpenTelemetry automatic instrumentation started successfully
2025-06-01T13:20:18.486-03:00  (node:9) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use a userland alternative instead.
2025-06-01T13:20:19.080-03:00  2025-06-01T16:20:19.080Z undefined WARN Failed extracting version /var/task
2025-06-01T13:20:19.092-03:00  EXTENSION Name: cloudwatch_lambda_agent State: Ready Events: [INVOKE, SHUTDOWN]
2025-06-01T13:20:19.092-03:00  START RequestId: fb9d5da4-5def-4fef-bc2d-4436e619b73a Version: $LATEST
2025-06-01T13:20:21.869-03:00  2025-06-01T16:20:21.869Z fb9d5da4-5def-4fef-bc2d-4436e619b73a INFO SQS response: { '$metadata': { httpStatusCode: 200, requestId: '2ba01a60-6091_

2025-06-01T16:20:21.869Z      fb9d5da4-5def-4fef-bc2d-4436e619b73a     INFO     SQS response: {
  '$metadata': {
    httpStatusCode: 200,
    requestId: '2ba01a60-6091-5c96-96c7-7ad6e0947fba',
    extendedRequestId: undefined,
    cfId: undefined,
    attempts: 1,
    totalRetryDelay: 0
  },
  MD5OfMessageAttributes: '0645f6d4012b32caf793dcafcc5545b7',
  MD5OfMessageBody: '4b9d86826c4f4076bf990e4e5a8972cd',
  MessageId: 'a1056443-b682-4a6b-a500-19aa2e8dbb40',
  SequenceNumber: '18894435532783343616'
}
2025-06-01T13:20:23.327-03:00  END RequestId: fb9d5da4-5def-4fef-bc2d-4436e619b73a
```

11. Application working

User can only upload a single file or a Github link. Repository should not exceed the size of 25 MB. Users are only allowed to upload 1 file or link per session.



Code Assistant

The code is about a binary search algorithm. It is an iterative version of the algorithm.

Insight:
The algorithm is a good example of a divide and conquer algorithm. It is a simple and efficient way to search for an element in a sorted array.

Context:
The code is part of a Java class that contains a binary search method. The method takes an array of integers and a target integer as input and returns the index of the target element in the array if it is found, or -1 if it is not found. The method uses a while loop to repeatedly divide the array into two halves until the target element is found or the array is empty.

The code also contains a main method that demonstrates the use of the binary search method. The main method creates an array of integers and a target integer, and then calls the binary search method to search for the target element in the array. If the element is found, the method returns the index of the element in the array. If the element is not found, the method returns -1.

The code is well-structured and easy to understand. The method is well-documented, and the comments explain the purpose of the method and the variables used in the method. The code is also well-formatted and easy to read.

The code is not optimized for performance, but it is efficient enough for most practical purposes. The algorithm is simple and easy to understand, and it is well-suited for searching for an element in a sorted array.

The code is not very flexible, as it is hardcoded to search for an element in a sorted array. However, the code can be easily modified to search for an element in a different data structure, such as a linked list or a hash table.

The code is not very scalable, as it is not optimized for large arrays.

12. Why?

I used SQS to decouple the front end from the back end to simplify debugging. Another reason was security. Since the API Gateway lacks proper authorization, allowing the front end to directly access the back end would expose it to potential threats. Due to decoupling, the back end doesn't need to handle incoming requests immediately, and the requests are guaranteed to be processed. Without this, high CPU utilization upon receiving requests could prevent them from being processed.

I used DynamoDB to store the userId to connectionId mapping because the application requires frequent read and write access. Using RDS would introduce unnecessary overhead, as each query goes through a SQL engine and maintains a TCP connection with each client, which limits scalability due to connection constraints. Since the data is simple and non-relational, a NoSQL key-value store like DynamoDB is a better fit for this use case.

13. Is it Well-Architected?

I have used AWS Managed service DynamoDB so that I don’t need to manage the database myself. I have also implemented proper monitoring logs and dashboard so that a failure in any system would be visible. My front-end code can be changed independently of the back-end as both are decoupled using a queue service. The auto scaling EC2 instances fetch code from the source code bucket so manual configuration of code and the environment is not required. Front-end also changes as per the code changes in the S3 bucket. All of these ensures operational excellence. Usage of CloudFront provides performance efficiency.

I have used roles with least privileges which ensures security of the architecture. A back-end failure would notify me via email, and all the components scale automatically, hence reliability is also ensured.

AWS cost optimizer has been enabled which would recommend me settings for cost optimization as well.

14. Security Measures

All the services in this architecture uses specific roles that provide least privilege access. Back-end computation is done in the EC2 instances deployed inside the Private subnet with no inbound access from the internet. Also, the front-end and the back-end has been decoupled by using SQS service. I have used NAT gateway so that EC2 can send messages to connected clients and download files from Github.

15. Model Selection

Microsoft's codebert-base model was used to create embeddings of query and user uploaded data because the model is trained on code and natural language to create better embeddings.

CodeLlama 7B Instruct model is used to answer the user queries. This model has been trained on code and natural languages and is fine tuned to support instruction handling. I didn't make any changes in the model. I deployed the model as it was.

16. Cost Optimization strategies

Each stored object is deleted after a day from S3 using lifecycle rul. This reduces cost because of less storage usage.

EC2 auto scaling with dynamic policies allows EC2 instances to shut down and start up whenever necessary. Hence, there is no manual reservation of EC2 instances.

17. Lesson learned and Future Improvements

Even though the model works reasonably well, if the embeddings aren't good enough and if the code chunks aren't properly embedded, the generated response could become irrelevant. In future, a better embedding model for query, and a better method for chunking user data could be used so that the context is relevant to the query and the LLM can answer the question. Currently, each file has its own chunk, and the embeddings aren't as close to the chunk's embedding as required for better performance. This causes the top_k result to vary, and therefore the results.