Phase 1

Contents

Team Information

id	name	email
22101164	Merna Adel Abdelrahman	merna.ibrahim.2023@aiu.edu.eg
22101332	Remonda Rezq Azer	remonda.rezq.2023@Aiu.edu.eg
22101137	Abdallah Ahmed Mahmoud Lasheen	abdalla.lasheen.2023@Aiu.edu.eg
22101053	Antonius sameh	antonius.mosad.2023@Aiu.edu.eg
22100933	Noura adel abd el Rahman Mohamed	nora.mohamed.2023@Aiu.edu.eg
22100665	Nourhan Abdelhamid Ahmed Mohammed	nourhan.mohammed.2023@Aiu.edu.eg

1. IAM

1 Identity and Access Management (IAM)

1. Purpose s Strategy

The primary goal of this module is to secure access to the Cloud-Based Learning Platform resources. Our design philosophy follows the **Principle of Least Privilege**, ensuring strict isolation between the microservices as required in the project specifications.

Key Design Decisions:

- **Service Isolation:** Instead of a generic EC2 role, we created distinct IAM roles for each microservice (e.g., Quiz-Service-Role, TTS-Service-Role). This ensures that if the TTS service is compromised, the attacker cannot access the Quiz or Chat data.
- **Container Security:** We implemented IAM Roles for Service Accounts (IRSA) to map permissions directly to the container tasks rather than the underlying EC2 instances.
- **Admin Security:** We enforced strict Multi-Factor Authentication (MFA) for all users with administrative privileges.

2. Implementation Steps

We executed the following steps to meet the requirements listed in Section 2.1:

1. **User Group Creation:**
    a. Platform-Admins: Granted AdministratorAccess with a strictly attached Force_MFA policy.

b. Backend-Devs: Granted access to development environments only, with restrictions on production S3 buckets.

2. **Role Configuration:**
    a. **Quiz-Generator-Role:** Configured for the Quiz Service to access quiz-service-storage-dev.
    b. **Document-Reader-Role:** Configured for the Document Reader Service to access document-reader-storage-dev.
    c. **Lambda-Cleanup-Role:** Created for the serverless maintenance functions.
3. **Policy Definition:** We authored custom JSON policies to restrict access to specific S3 buckets and RDS instances per service.

3. Configurations (Policy JSONs)

A. Quiz Service Policy

Rationale: This policy grants the Quiz Service permission to store generated quizzes and read processed document notes, as per the isolation requirements.

{ "Version": "2012-10-17", "Statement": [{ "Sid": "QuizBucketAccess", "Effect": "Allow", "Action": ["s3:PutObject", "s3:GetObject"], "Resource": "arn:aws:s3:::quiz-service-storage-dev/*" // Access limited strictly to the Quiz bucket }, { "Sid": "ReadProcessedNotes", "Effect": "Allow", "Action": "s3:GetObject", "Resource": "arn:aws:s3:::document-reader-storage-dev/notes/*" }, { "Sid": "ConnectToQuizDB", "Effect": "Allow", "Action": "rds-db:connect", "Resource": "arn:aws:rds-db:us-east-1:123456789:dbuser:quiz_db/app_user" }] }

B. Admin MFA Enforcement Policy

Rationale: This policy explicitly denies all actions for Admin users unless they are authenticated via MFA.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAllExceptMFA",
      "Effect": "Deny",
      "NotAction": [
        "iam:CreateVirtualMFADevice",
        "iam:EnableMFADevice",
        "iam:ResyncMFADevice",
        "iam:ListMFADevices"
```

```
      ],
      "Resource": "*",
      "Condition": {
        "BoolIfExists": {
          "aws:MultiFactorAuthPresent":  "false"
        }
      }
    }
  ]
}
```

4. Access Control Matrix

The following table illustrates the permission boundaries established for the project resources:

Entity (Role/User)	Target Resource	Access Level	Rationale
Admin User	All Resources	Full Access	System Administration (MFA Required)
TTS-Service-Role	tts-service-storage-dev	Read/Write	Audio file generation C storage [12]
STT-Service-Role	stt-service-storage-dev	Read/Write	Uploading audio for transcription [13]
Chat-Service-Role	chat-service-storage-dev	Read/Write	Storing conversation history [14]
Quiz-Service-Role	quiz-service-storage-dev	Read/Write	storing quiz templates C results [15]
Lambda-Cleanup	All Buckets (Lifecycle)	Delete Only	Automated file archiving/cleanup [16]

5. Deliverables Included

In accordance with the project requirements, the following files are attached in the IAM_Deliverables/ folder :

1. iam_role_definitions.pdf: Detailed documentation of all created roles.

2. policies/: Folder containing the raw .json policy files.
3. access_control_matrix.xlsx: Comprehensive permission matrix.
4. infrastructure/iam.tf: Terraform scripts used for provisioning the IAM layer.
5. screenshots/: Evidence of User creation, Group assignment, and MFA activation status.

2s3.Compute (EC2) s Storage (EBS) Layer

1. Overview

In this phase, I provisioned the compute and storage resources required to host themicroservices, Kafka cluster, and Zookeeper ensemble. The implementation focuses onHigh Availability (HA) by distributing resources across two Availability Zones (us-east-1a,us-east-1b) and ensuring Data Security through full encryption.

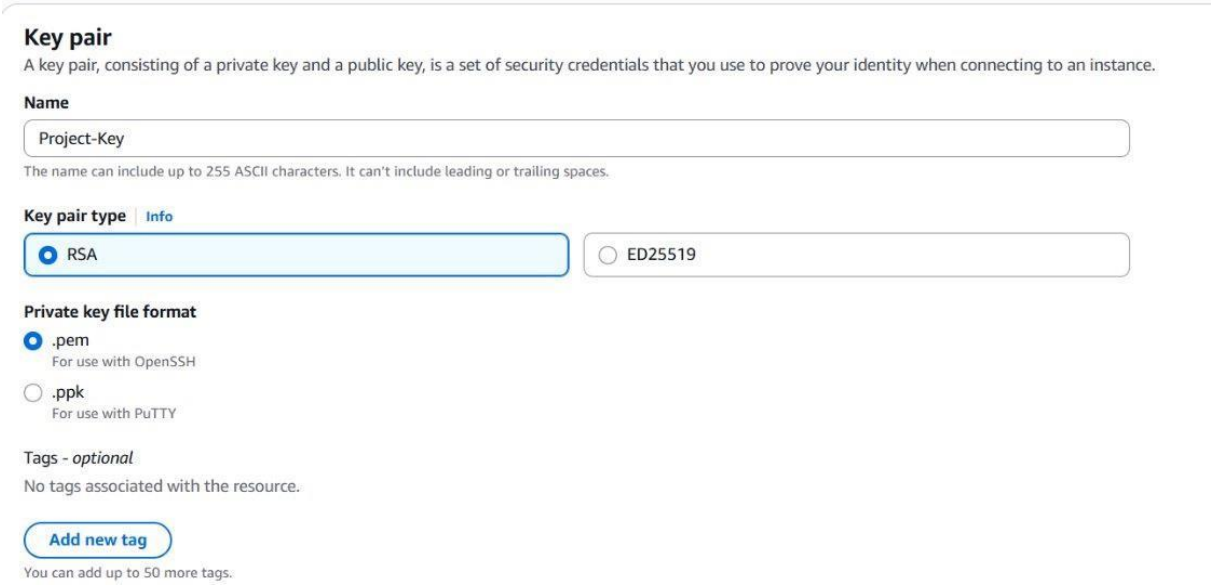
2.1 SSH Key Pair Configuration

To facilitate secure remote access (SSH) to the EC2 instances for management and troubleshooting purposes, adedicated RSA key pair was generated prior to launching the infrastructure.

Configuration Details:

- Name: **Project-Key**
- Type: **RSA (2048-bit)** – Chosen for standard security compatibility.
- Format: **.pem (Privacy Enhanced Mail)** – Selected for compatibility with OpenSSH clients and Linux/Macenvironments.

This key is securely stored and is associated with all Launch Templates (Container-Host, Kafka-Broker, andZookeeper-Node) to allow administrative access if required.

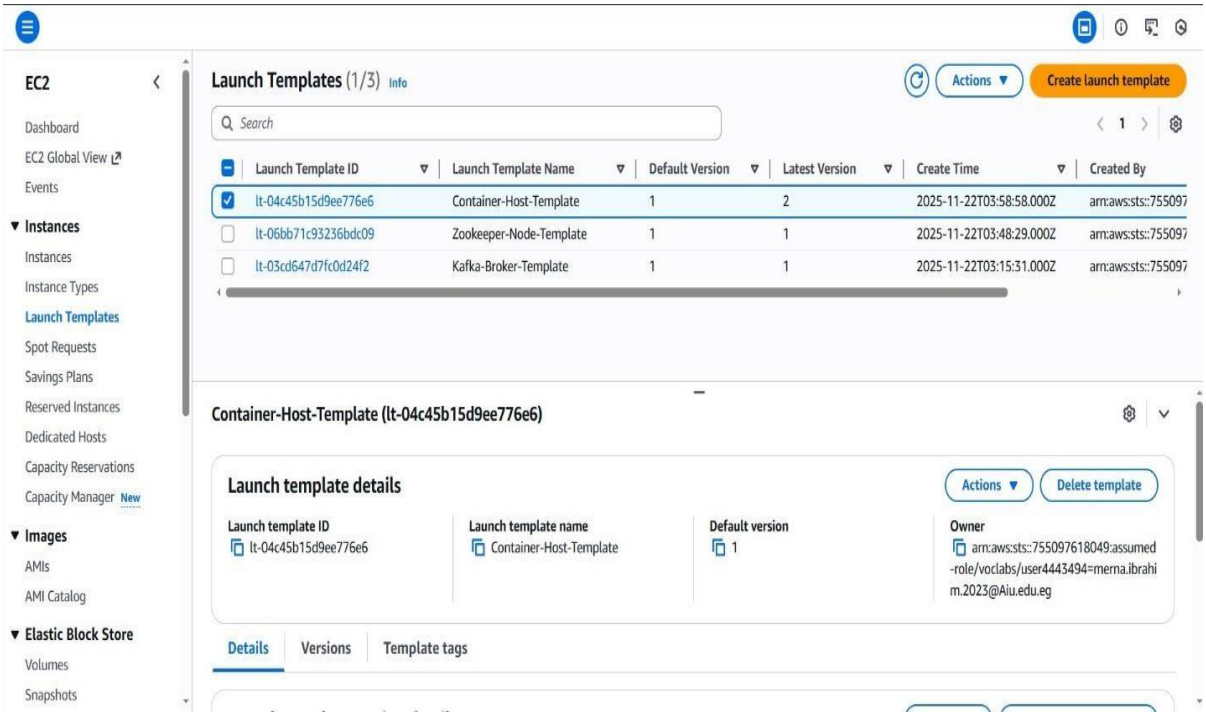


2.2 Launch Templates Configuration

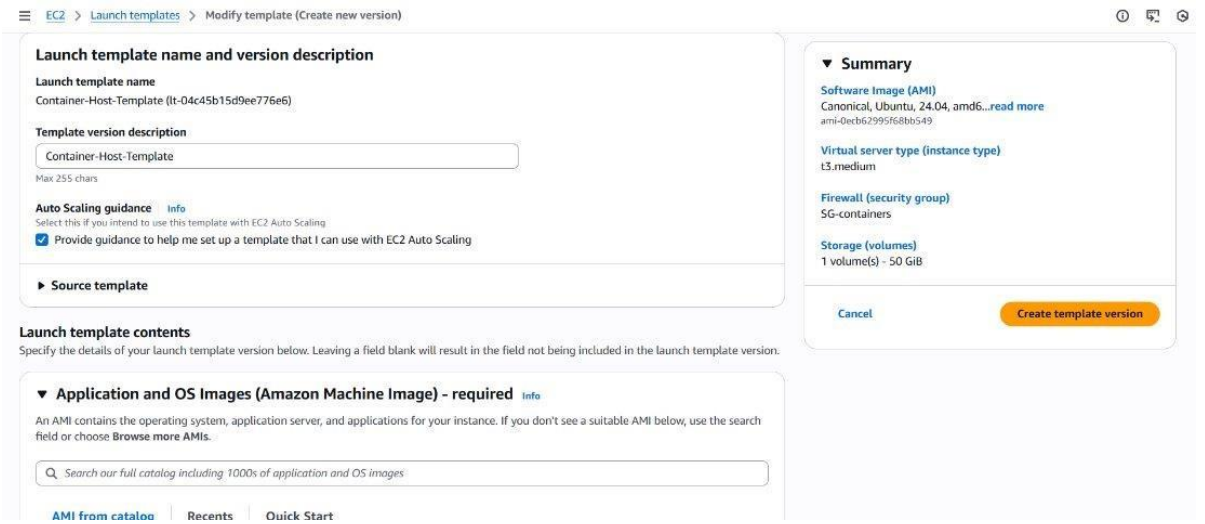
To standardize the deployment of the microservices and message brokers, I created three distinct Launch Templates. These templates pre-define the Amazon Machine Image (Ubuntu Server 24.04 LTS), Instance Type (t3.medium), and network security settings, ensuring consistency across the environment.

Created Templates:

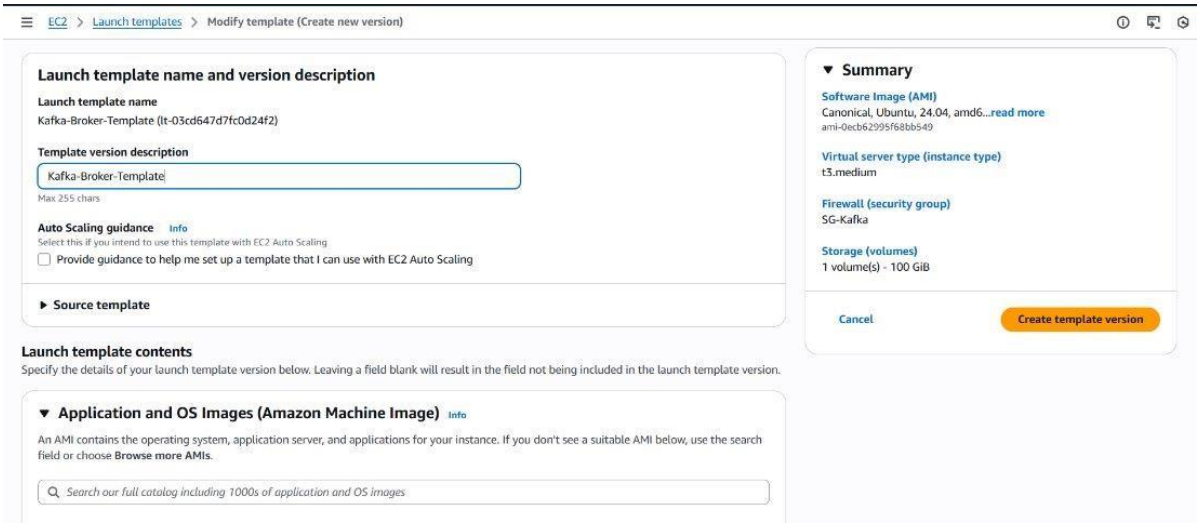
- Container-Host-Template: For the Docker/Kubernetes cluster.
- Kafka-Broker-Template: For the Kafka streaming cluster.
- Zookeeper-Node-Template: For the coordination ensemble.



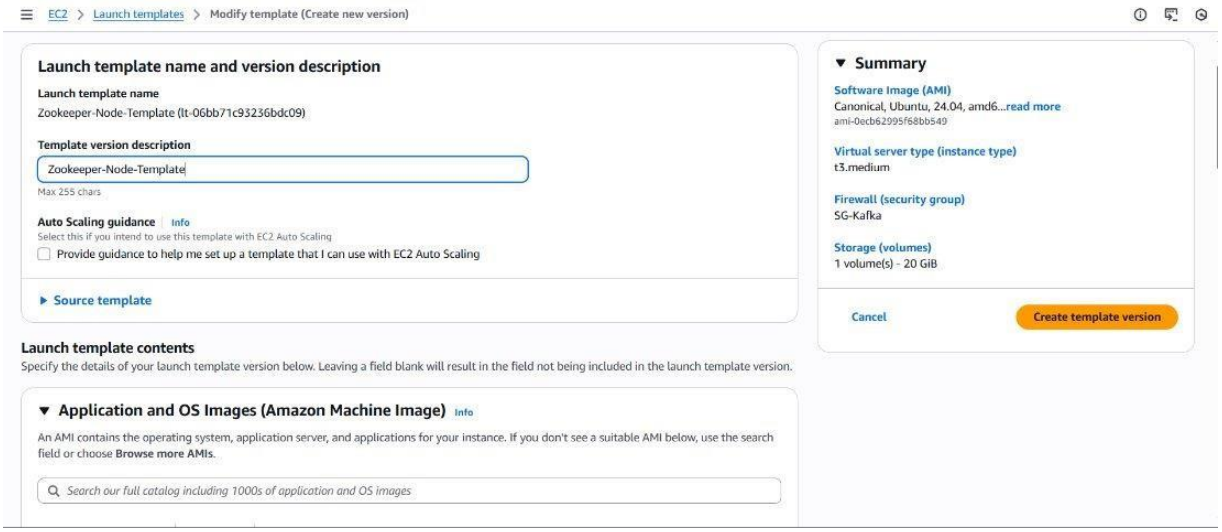
Create Container-Host-Template



Create Kafka-Broker-Template



Create Zookeeper-Node-Template



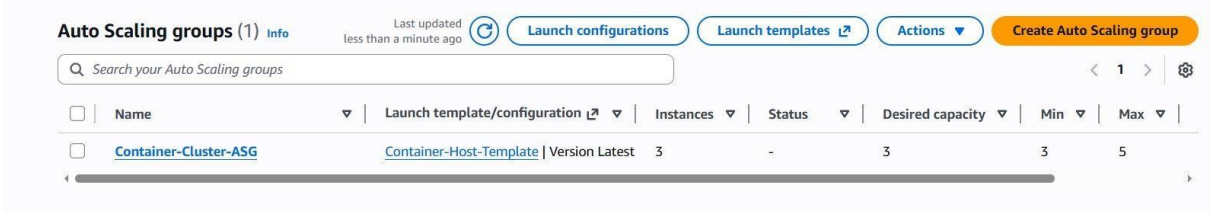
2.3 Auto Scaling Group (Container Cluster)

- For the application layer, I implemented an Auto Scaling Group (Container-Cluster-ASG) to ensure scalability and fault tolerance. The group isconfigured to launch instances into Private Subnets across two AvailabilityZones (us-east-1a, us-east-1b).

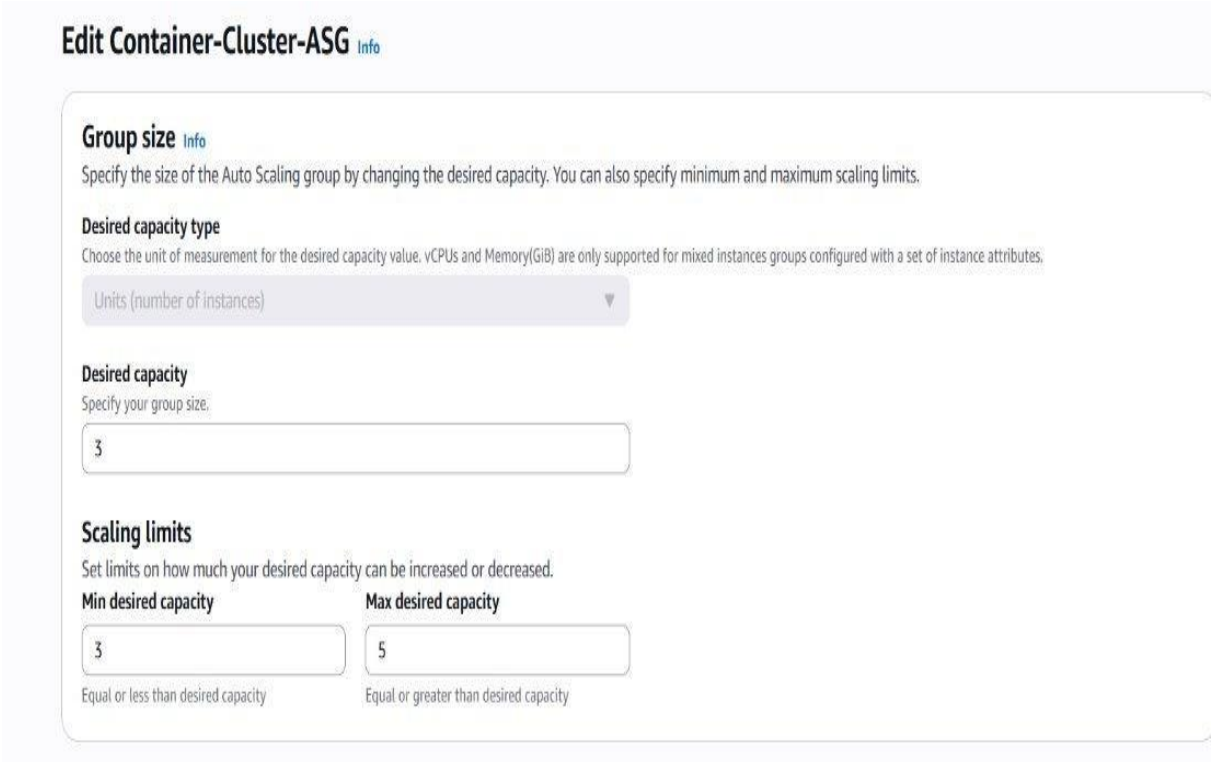
Configuration Details:

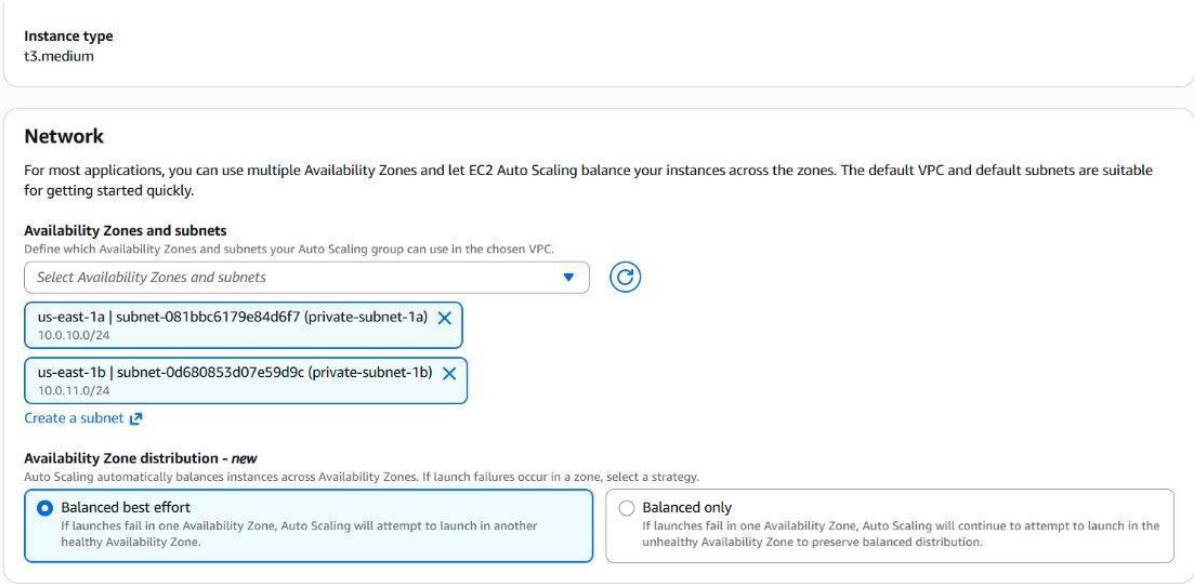
- Desired Capacity: 3 instances.
- Minimum Capacity: 3 instances.
- Maximum Capacity: 5 instances.

Health Checks: EC2-based health checks are enabled.



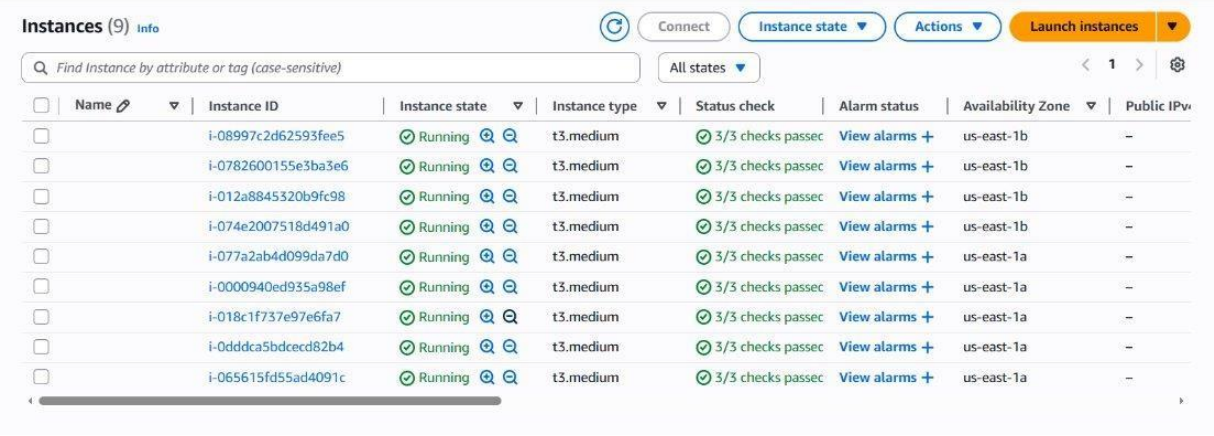
Create Container-Cluster-ASG

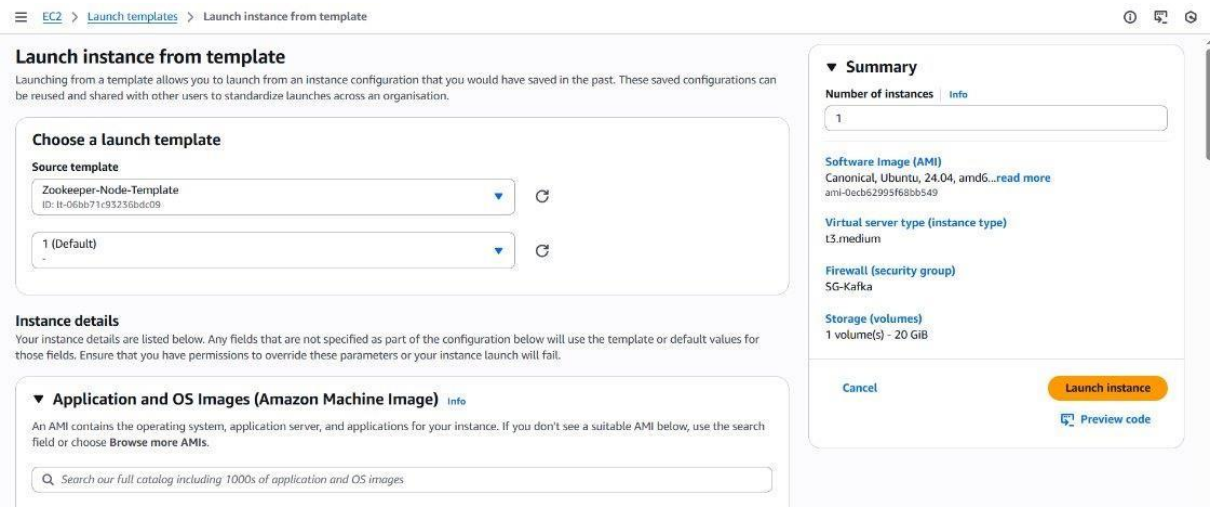




2.4 Instances s High Availability (Multi-AZ)

- A total of 9 EC2 instances were successfully deployed. To achieve High Availability (HA) as per thearchitecture requirements, the Kafka and Zookeeper brokers were manually distributed across distinctAvailability Zones.
- Distribution:
- Containers: Automatically balanced by ASG.
- Kafka/Zookeeper: 1 nodes in us-east-1a and 2 node in us-east-1b.





5. EBS Storage s Encryption

Persistent storage was provisioned using GP3 EBS volumes attached to all instances. To comply with strict

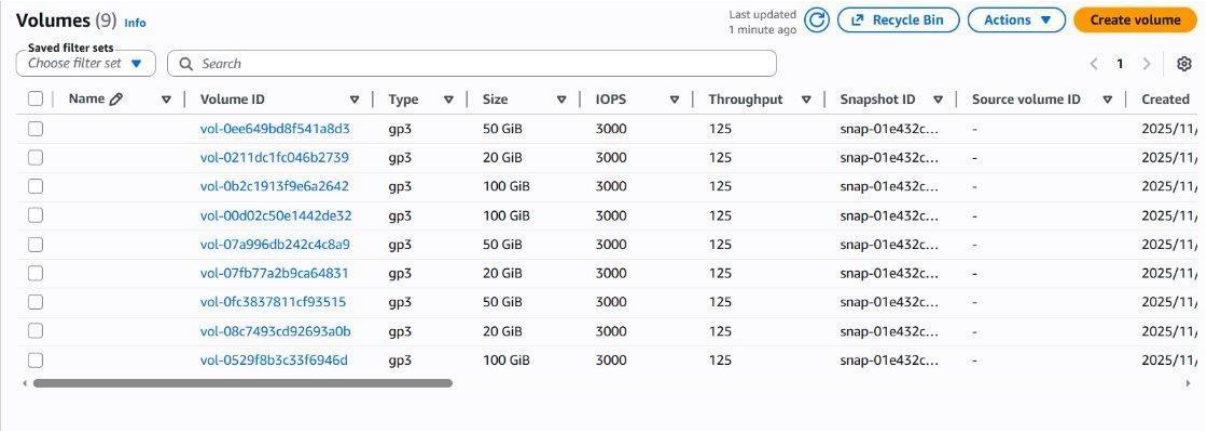
security requirements, Server-Side Encryption (AWS KMS) was enabled for all volumes holding sensitive

data.

Volume Specifications:

- Kafka Brokers: 100 GiB per instance.
- Container Hosts: 50 GiB per instance.
- Zookeeper Nodes: 20 GiB per instance.

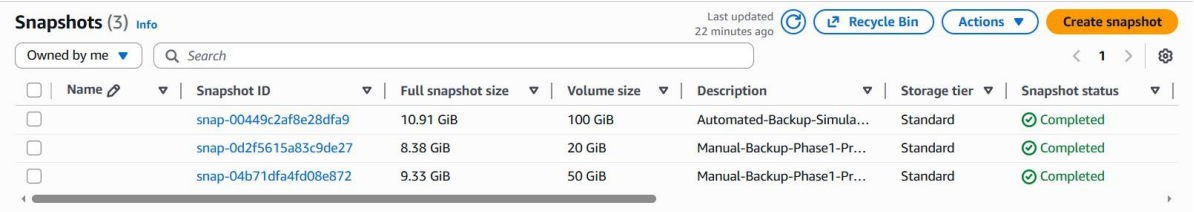
Service Tier	Size	Type	Encryption	Mount Point
Kafka Brokers	100 GiB	gp3	Encrypted	/dev/sda1
Container Hosts	50 GiB	gp3	Encrypted	/dev/sda1
Zookeeper Nodes	20 GiB	gp3	Encrypted	/dev/sda1



6. Snapshot Schedule and Retention Policy

- A comprehensive backup strategy was designed using Amazon Data Lifecycle Manager(DLM) to automate the creation and retention of EBS snapshots.
- Policy Specifications:
- Schedule Name: Project-Daily-Backup
- Target Resource: Instances tagged with Project: Phase1
- Frequency: Daily at 00:00 UTC.
- Retention Rule: Keep the last 7 snapshots (Weekly rotation) to optimize costs whileensuring data availability.
- Implementation Note: Due to lab account permissions restricting thedlm:GetLifecyclePolicies action, the automated policy could not be fully

activated.As a Proof of Concept (PoC), a manual snapshot was successfully created and verifiedfor the critical Kafka data volumes.



7. Disaster Recovery Procedures

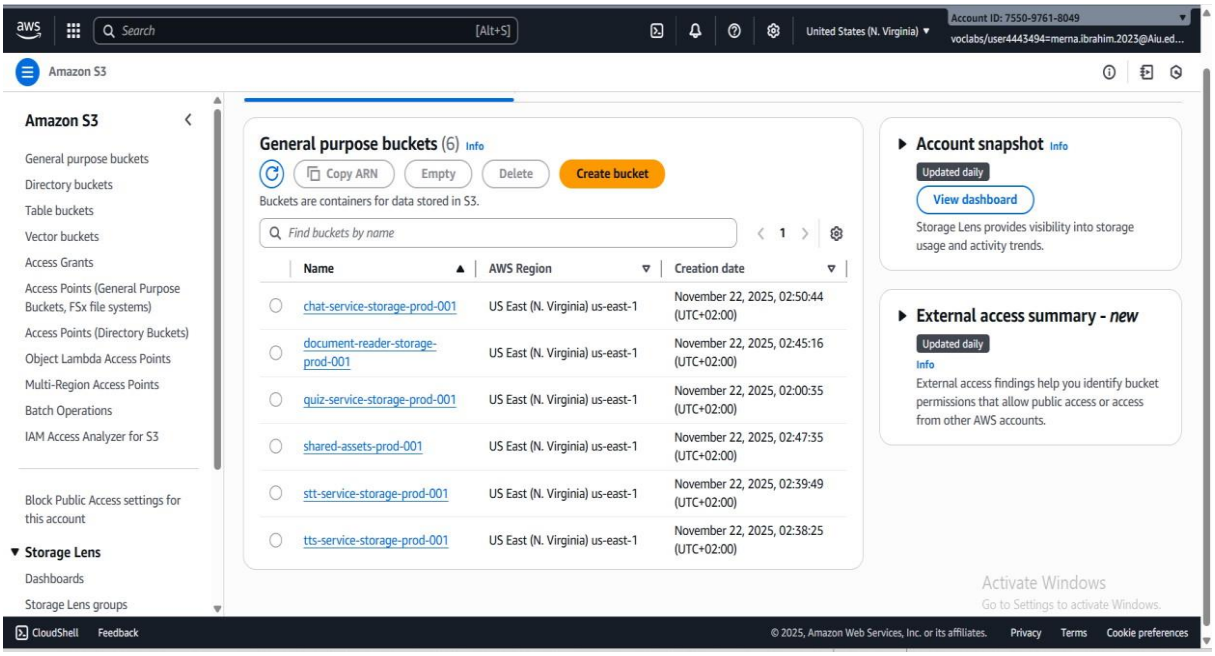
- Objective Recover data from EBS volume failures using manual snapshots createdin Phase 1.
- Recovery Procedures
- Identify Failure:
- Detect instance failure via CloudWatch (StatusCheckFailed) or disk I/O errors in logs.
- Stop Instance:
- Stop the affected EC2 instance (do *not* terminate) to freeze data state.
- Note the volume mount point (e.g., /dev/sda1).
- Restore from Backup:
- Locate the latest snapshot tagged Manual-Backup-Proof.
- Create a new volume from the snapshot in the same Availability Zone as the instance.
- Ensure volume settings match (Type: gp3, Encryption: Enabled).
- Swap C Verify:
- Detach the corrupted volume and attach the new one using the correct mount point.

Start the instance and verify service health via SSH.

4. Simple Storage Service (S3)

1. S3 Bucket Naming Convention Document: confirms the naming pattern used for all S3 buckets.

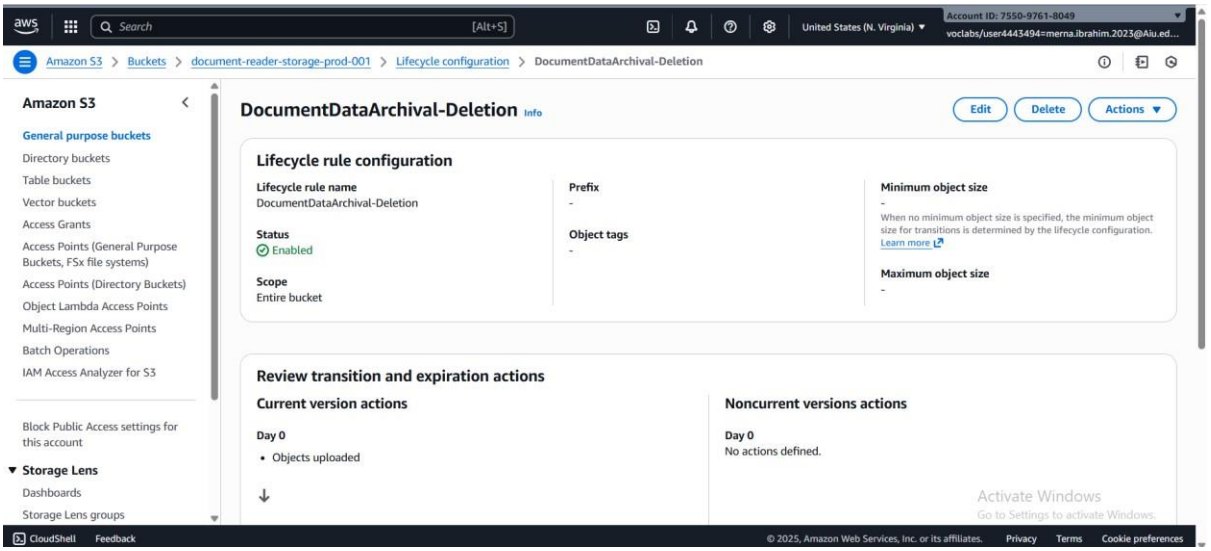
Document Action	Details to be Included
Document Name	S3_Naming_Convention_Document
Naming Pattern	{service}-service-storage-{env}-001
Example Buckets	tts-service-storage-prod-001, stt-service-storage-prod-001, chat-service-storage-prod-001, document-reader-storage-prod-001, quiz-service-storage-prod-001, shared-assets-prod-001

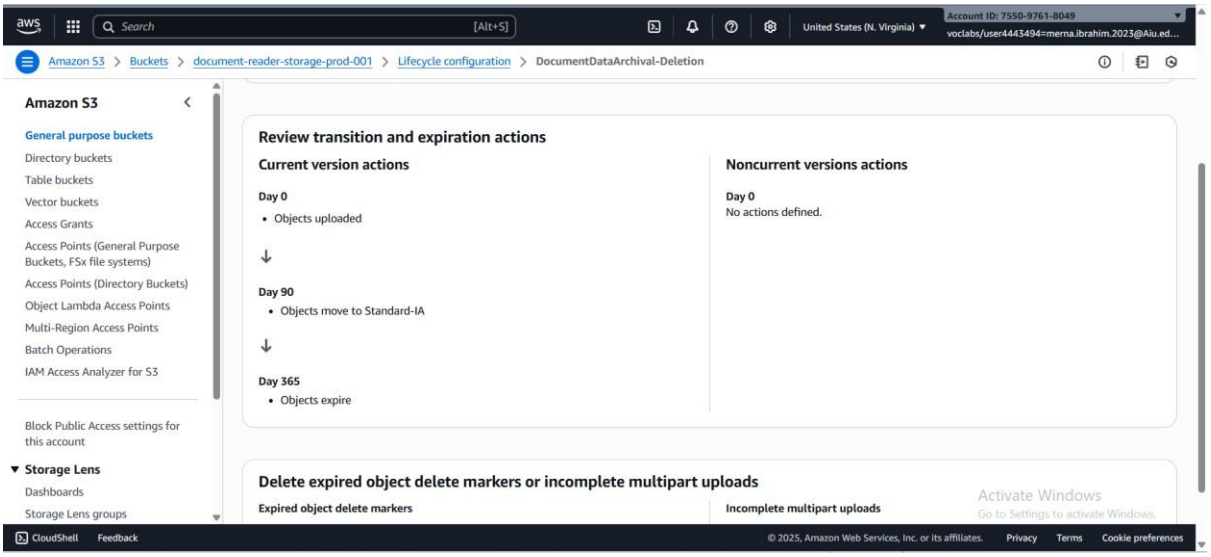


2. Lifecycle Policy Configurations

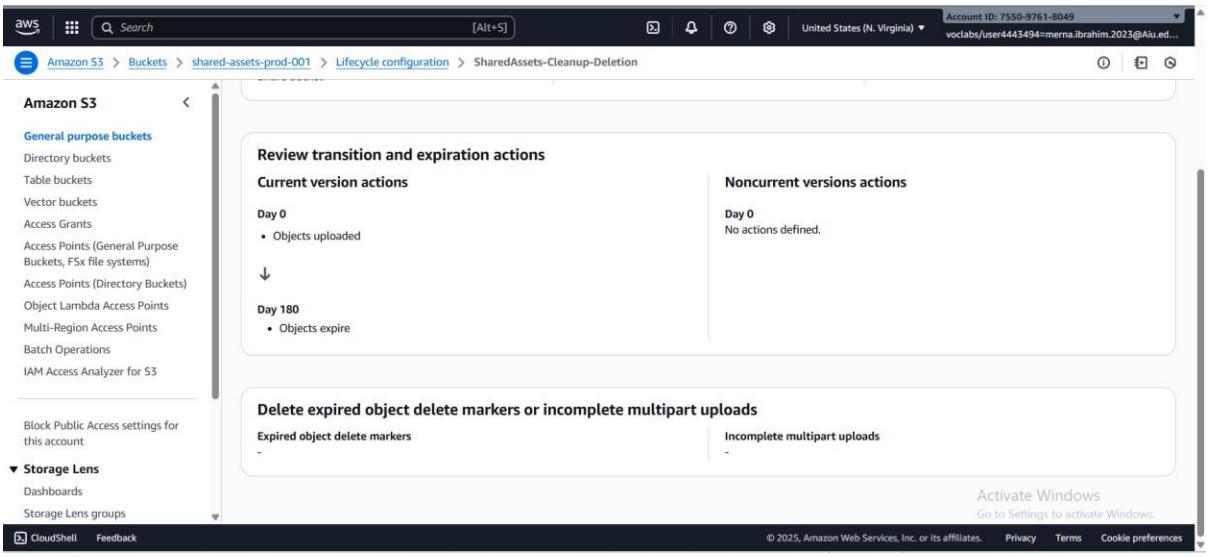
Bucket Name	Rule Name	Transition	Expiration	Justification
Quiz / Chat / Document	ChatHistory-Archival-Deletion	After 90 days → Standard-IA	After 365 days	Critical data with 1-year retention requirement.
TTS / STT	TTS-Audio-Archival-Deletion	After 60 days → Standard-IA	After 180 days	Temporary files, deleted quickly for cost savings.
Shared Assets	SharedAssets-Cleanup-Deletion	None (remains Standard)	After 180 days	Static assets remain available; old backups removed.

document-reader-storage-prod-001

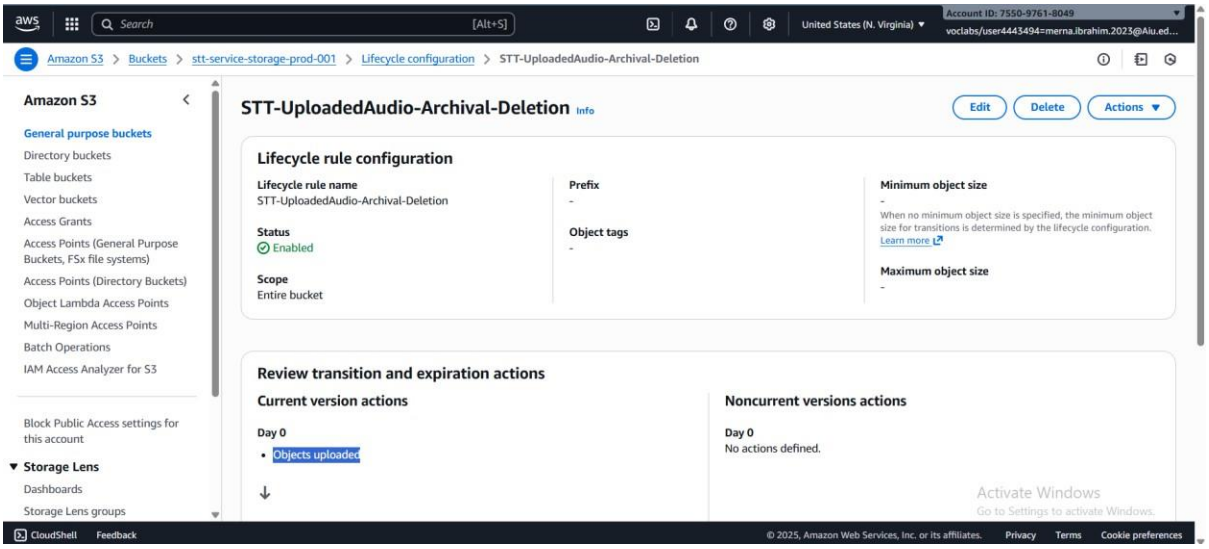


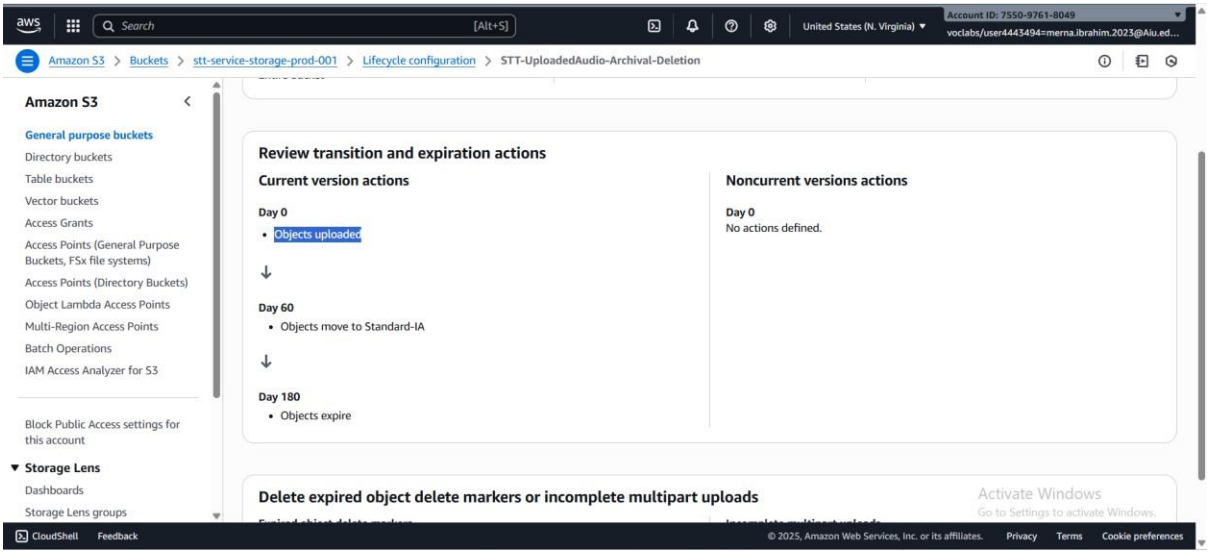


shared-assets-prod-001

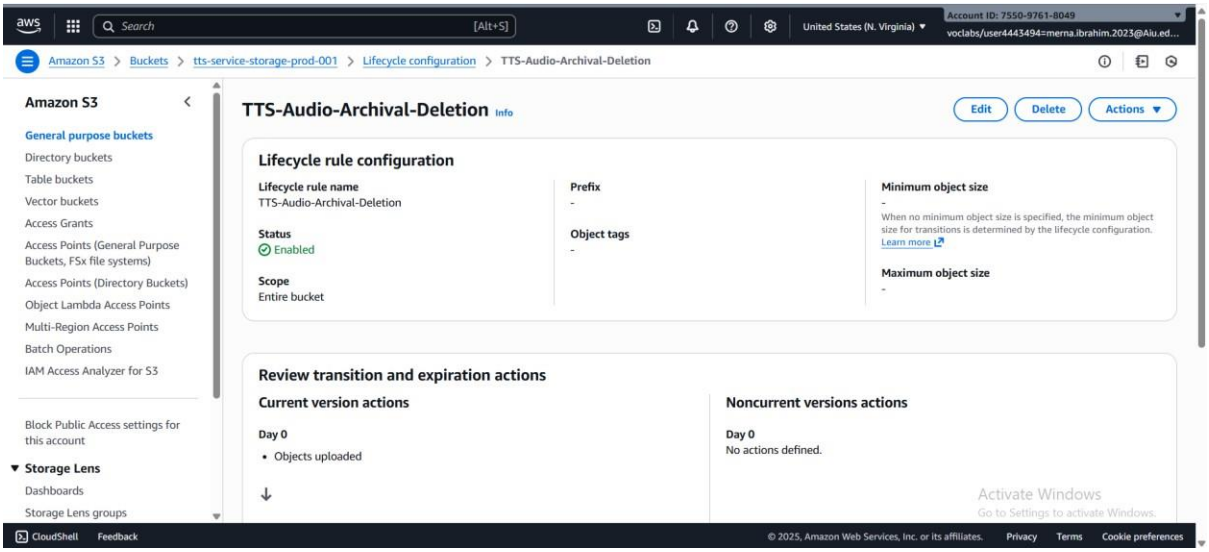


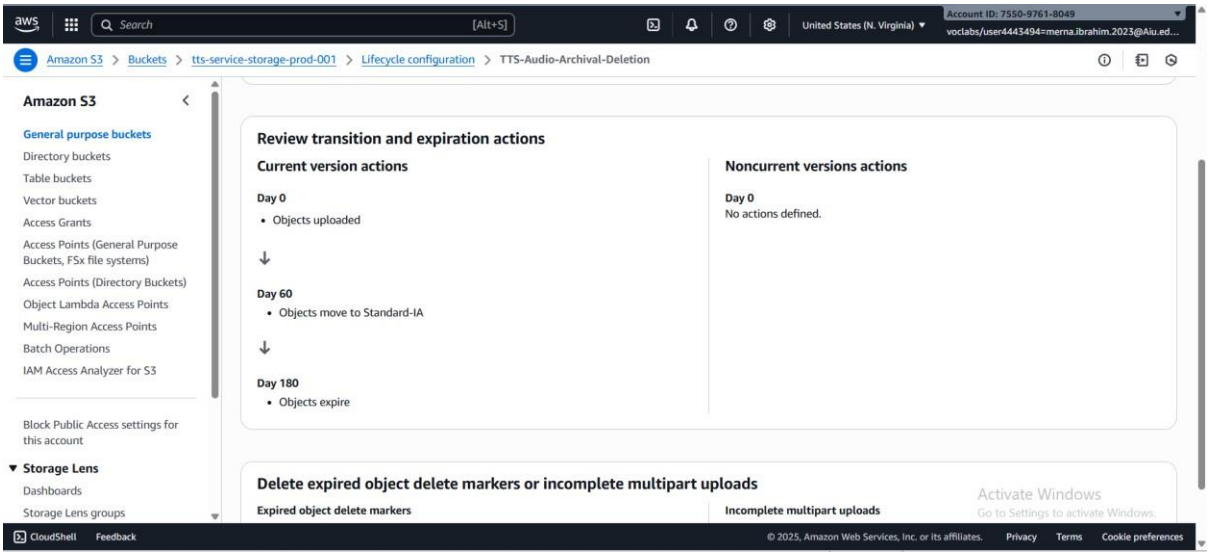
stt-service-storage-prod-001



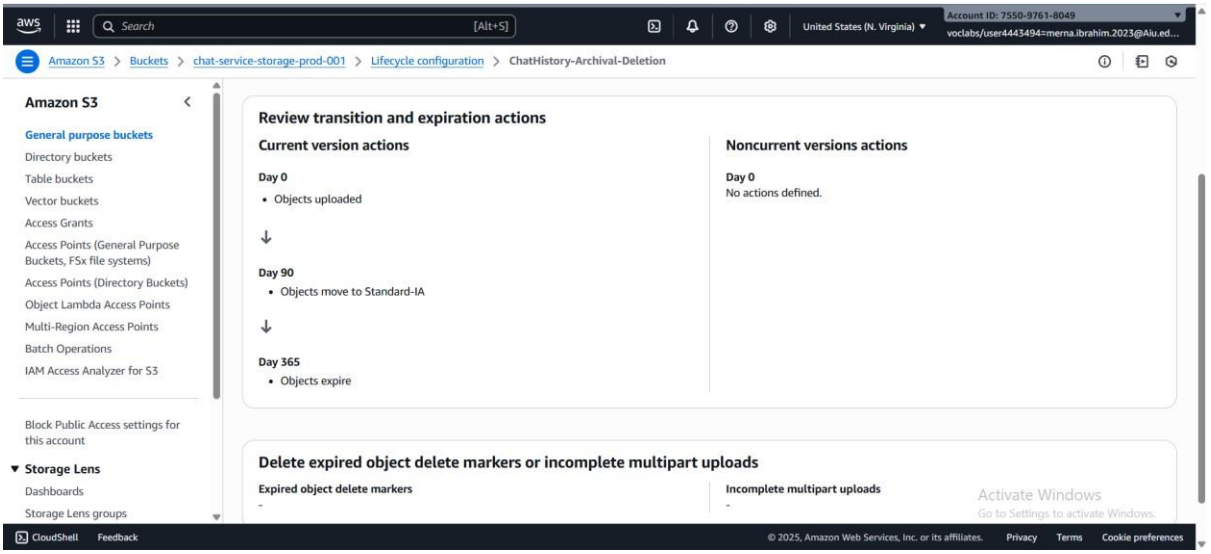


tts-service-storage-prod-001

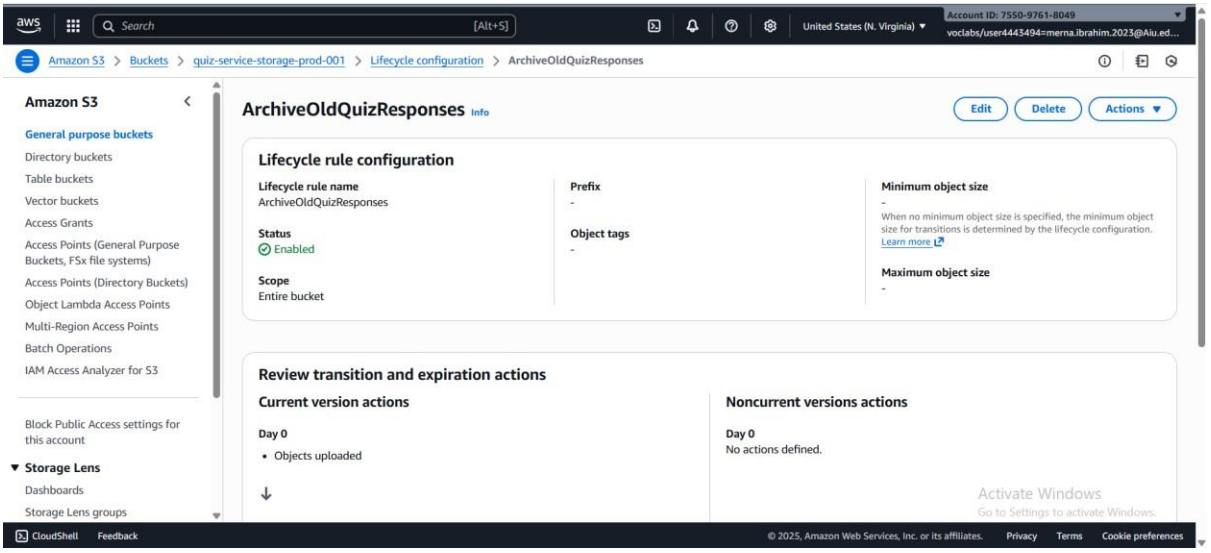


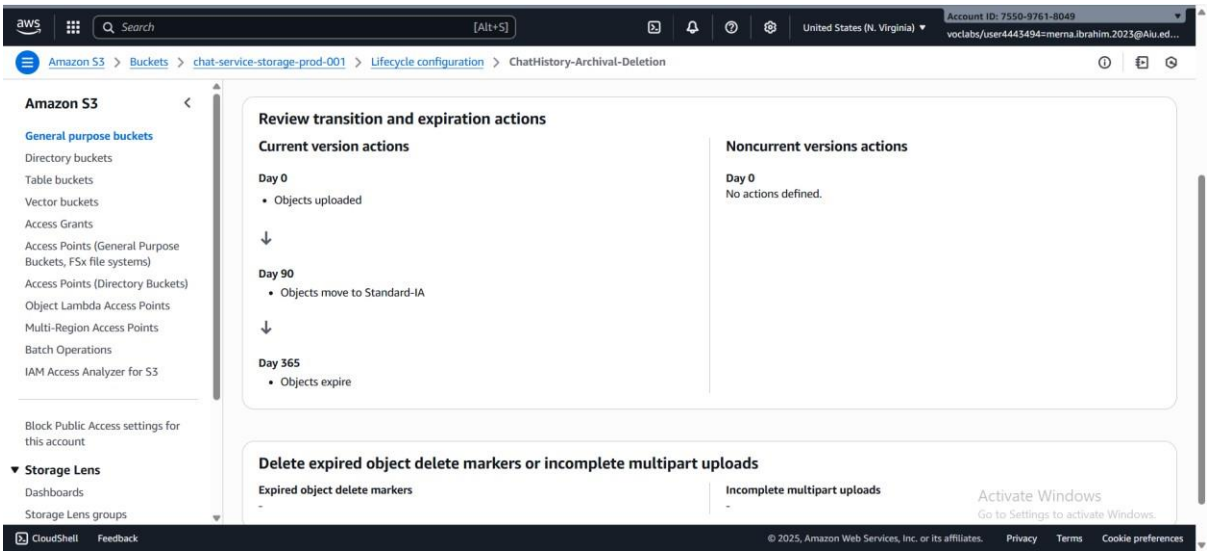


chat-service-storage-prod-001



quiz-service-storage-prod-001





Bucket Policy JSON Documentation

- This section provides the final JSON code used to implement the Principle of Least Privilege and ensure Isolated Storage for each of the six S3 buckets.

Quiz Service Bucket Policy: IAM Role: QuizServiceRole Purpose: Restricts access to the quiz-service-storage-prod-001 bucket to only the Quiz Service Role.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AllowAccessOnlyForQuizServiceRole",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::[ACCOUNT-ID]:role/QuizServiceRole"
   },
   "Action": "s3:*",
   "Resource": [
    "arn:aws:s3:::quiz-service-storage-prod-001",
    "arn:aws:s3:::quiz-service-storage-prod-001/*"
   ]
  },
  {
   "Sid": "DenyAllOthers",
   "Effect": "Deny",
```

```
      "Principal": "*",
      "Action": "s3:*",
      "Resource": [
       "arn:aws:s3:::quiz-service-storage-prod-001",
       "arn:aws:s3:::quiz-service-storage-prod-001/*"
      ],
      "Condition": {
       "StringNotLike": {
        "aws:PrincipalArn": "arn:aws:iam::[ACCOUNT-ID]:role/QuizServiceRole"
       }
      }
     }
    ]
}
```

2. TTS Service Bucket Policy: IAM Role: TtsServiceRole Purpose: Restricts access to the tts-service-storage-prod-001 bucket to only the TTS Service Role.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AllowAccessOnlyForTtsServiceRole",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::[ACCOUNT-ID]:role/TtsServiceRole"
   },
   "Action": "s3:*",
   "Resource": [
    "arn:aws:s3:::tts-service-storage-prod-001",
    "arn:aws:s3:::tts-service-storage-prod-001/*"
   ]
  },
  {
   "Sid": "DenyAllOthers",
   "Effect": "Deny",
   "Principal": "*",
```

```
      "Action": "s3:*",
      "Resource": [
       "arn:aws:s3:::tts-service-storage-prod-001",
       "arn:aws:s3:::tts-service-storage-prod-001/*"
      ],
      "Condition": {
       "StringNotLike": {
        "aws:PrincipalArn": "arn:aws:iam::[ACCOUNT-ID]:role/TtsServiceRole"
       }
      }
     }
    ]
}
```

3. STT Service Bucket Policy

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AllowAccessOnlyForSttServiceRole",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::[ACCOUNT-ID]:role/SttServiceRole"
   },
   "Action": "s3:*",
   "Resource": [
    "arn:aws:s3:::stt-service-storage-prod-001",
    "arn:aws:s3:::stt-service-storage-prod-001/*"
   ]
  },
  {
   "Sid": "DenyAllOthers",
   "Effect": "Deny",
   "Principal": "*",
   "Action": "s3:*",
```

```
      "Resource": [
       "arn:aws:s3:::stt-service-storage-prod-001",
       "arn:aws:s3:::stt-service-storage-prod-001/*"
      ],
      "Condition": {
       "StringNotLike": {
        "aws:PrincipalArn": "arn:aws:iam::[ACCOUNT-ID]:role/SttServiceRole"
       }
      }
     }
    ]
}
```

4. Chat Service Bucket Policy: IAM Role: ChatServiceRole Purpose: Restricts access to the chat-service-storage-prod-001 bucket to only the Chat Service Role.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AllowAccessOnlyForChatServiceRole",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::[ACCOUNT-ID]:role/ChatServiceRole"
   },
   "Action": "s3:*",
   "Resource": [
    "arn:aws:s3:::chat-service-storage-prod-001",
    "arn:aws:s3:::chat-service-storage-prod-001/*"
   ]
  },
  {
   "Sid": "DenyAllOthers",
   "Effect": "Deny",
   "Principal": "*",
   "Action": "s3:*",
   "Resource": [
```

```
        "arn:aws:s3:::chat-service-storage-prod-001",
        "arn:aws:s3:::chat-service-storage-prod-001/*"
      ],
      "Condition": {
       "StringNotLike": {
        "aws:PrincipalArn": "arn:aws:iam::[ACCOUNT-ID]:role/ChatServiceRole"
       }
      }
     }
    ]
}
```

5. Document Reader Bucket Policy:

- IAM Role: DocumentReaderRole
- Purpose: Restricts access to the document-reader-storage-prod-001 bucket to only the Document Reader Role.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AllowAccessOnlyForDocumentReaderRole",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::[ACCOUNT-ID]:role/DocumentReaderRole"
   },
   "Action": "s3:*",
   "Resource": [
    "arn:aws:s3:::document-reader-storage-prod-001",
    "arn:aws:s3:::document-reader-storage-prod-001/*"
   ]
  },
  {
   "Sid": "DenyAllOthers",
   "Effect": "Deny",
   "Principal": "*",
```

```
      "Action": "s3:*",
      "Resource": [
       "arn:aws:s3:::document-reader-storage-prod-001",
       "arn:aws:s3:::document-reader-storage-prod-001/*"
      ],
      "Condition": {
       "StringNotLike": {
        "aws:PrincipalArn": "arn:aws:iam::[ACCOUNT-ID]:role/DocumentReaderRole"
       }
      }
     }
    ]
}
```

6. Shared Assets Bucket Policy

IAM Role: SharedAssetsRole

Purpose: Restricts access to the shared-assets-prod-001 bucket to only the Shared Assets Role.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AllowAccessOnlyForSharedAssetsRole",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::[ACCOUNT-ID]:role/SharedAssetsRole"
   },
   "Action": "s3:*",
   "Resource": [
    "arn:aws:s3:::shared-assets-prod-001",
    "arn:aws:s3:::shared-assets-prod-001/*"
   ]
  },
  {
   "Sid": "DenyAllOthers",
   "Effect": "Deny",
```

```
        "Principal": "*",
        "Action": "s3:*",
        "Resource": [
          "arn:aws:s3:::shared-assets-prod-001",
          "arn:aws:s3:::shared-assets-prod-001/*"
        ],
        "Condition": {
          "StringNotLike": {
            "aws:PrincipalArn": "arn:aws:iam::[ACCOUNT-ID]:role/SharedAssetsRole"
          }
        }
      }
    ]
}
```

---

5. Virtual Private Cloud (VPC)

1. Purpose of the VPC

- Network isolation
- Traffic segmentation
- Controlled internet access
- Strong security boundaries for each tier

2. VPC Configuration Requirements

2.1 Create VPC

CIDR Block: 10.0.0.0/16

This CIDR allows up to 65,536 IP addresses, suitable for multi-tier distributed architecture.

2.2 Subnet Architecture

Subnet Tier	CIDR Block	Availability Zone
Public Subnet 1	10.0.1.0/24	AZ1
Public Subnet 2	10.0.2.0/24	AZ2
Private Subnet 1	10.0.10.0/24	AZ1
Private Subnet 2	10.0.11.0/24	AZ2

Data Subnet 1 (RDS)	10.0.20.0/24	AZ1
Data Subnet 2 (RDS)	10.0.21.0/24	AZ2
Kafka Subnet 1	10.0.30.0/24	AZ1
Kafka Subnet 2	10.0.31.0/24	AZ2

2.3 Internet Gateway Configuration

Attach an Internet Gateway (IGW) to the VPC to enable outbound and inbound internet traffic for public subnets.

2.4 NAT Gateway Configuration

Deploy one NAT Gateway per Availability Zone inside each public subnet. Private subnets route 0.0.0.0/0 traffic to NAT Gateways instead of the Internet Gateway.

2.5 Route Tables Setup

- Public route table → routes 0.0.0.0/0 to Internet Gateway
- Private route table → routes 0.0.0.0/0 to NAT Gateway
- Data subnets → no internet route (fully isolated)
- Kafka subnets → no internet route (internal cluster only)

2.6 Network ACLs

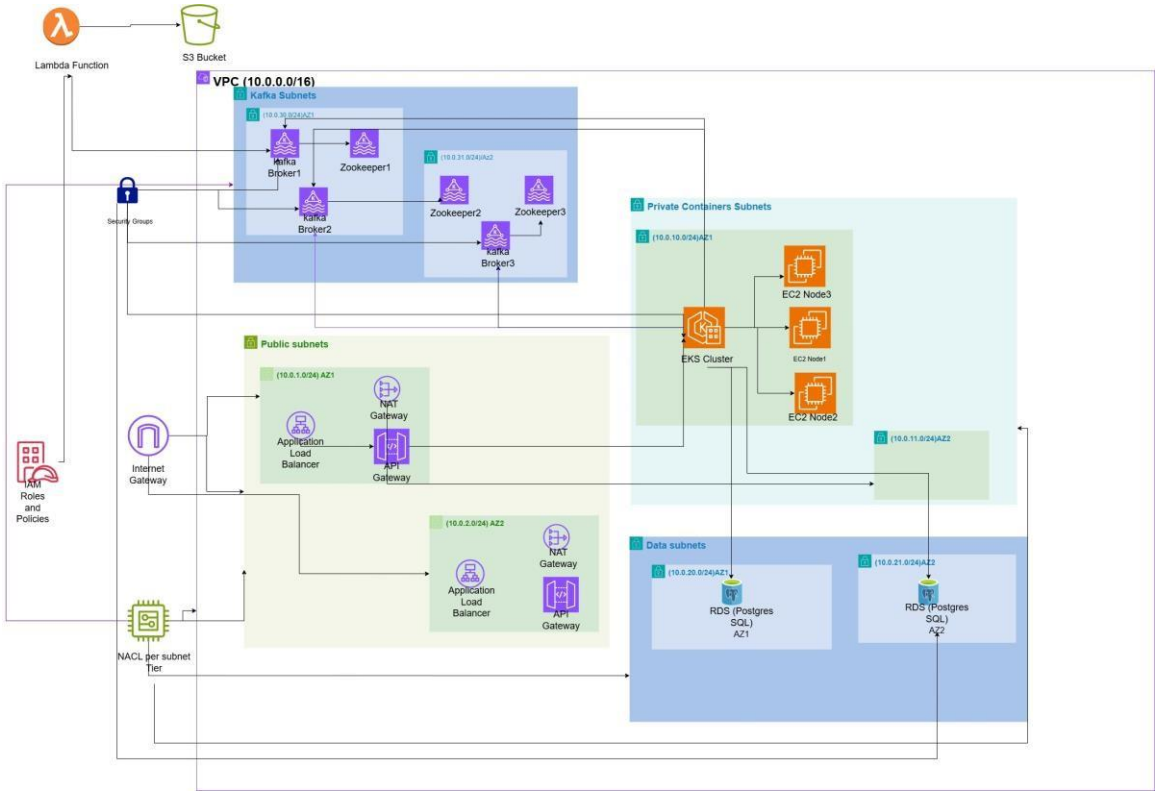
Network ACLs (NACLs) provide stateless packet filtering at the subnet level. Each subnet tier receives its own NACL for boundary-level protection.

2.7 Security Groups

Security Groups act as stateful firewalls for EC2, RDS, containers, load balancers, and Kafka brokers.

3. Deliverables

3.1 VPC Architecture Diagram



3.2 Subnet Allocation Table

Subnet Name	CIDR	AZ	Purpose
public-subnet-1a	10.0.1.0/24	us-east-1a	Load Balancers
public-subnet-1b	10.0.2.0/24	us-east-1b	Load Balancers
private-subnet-1a	10.0.10.0/24	us-east-1a	Containers
private-subnet-1b	10.0.11.0/24	us-east-1b	Containers
data-subnet-1a	10.0.20.0/24	us-east-1a	RDS Database
data-subnet-1b	10.0.21.0/24	us-east-1b	RDS Database
kafka-subnet-1a	10.0.30.0/24	us-east-1a	Kafka Cluster
kafka-subnet-1b	10.0.31.0/24	us-east-1b	Kafka Cluster

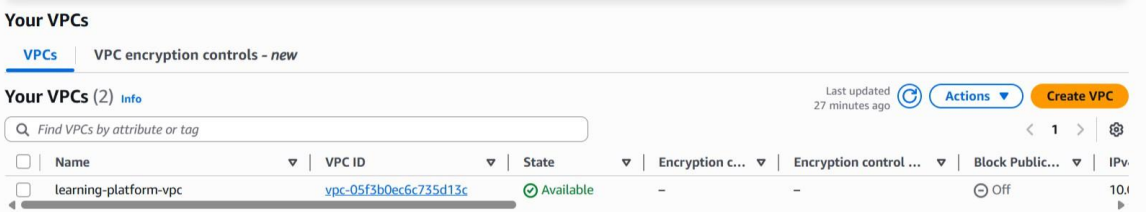
3.3 Security Group Definitions

Security Group	Inbound Rules	Outbound Rules	Description
SG-loadbalancer	HTTP 80, HTTPS 443 (0.0.0.0/0)	Allow all traffic	Allow Internet traffic to LB
SG-containers	TCP 3000, 8000 from SG-loadbalancer	Allow all traffic	App containers access
SG-RDS	PostgreSQL 5432 from SG-containers	Allow all traffic	DB access from containers
SG-Kafka	TCP 9092 from SG-containers	Allow all traffic	Kafka access from containers

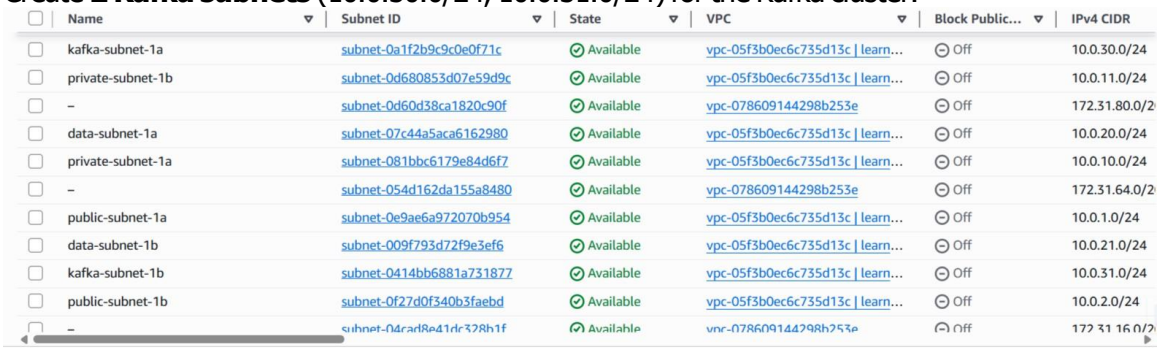
3.4 Network ACL Rules

NACL Name	Inbound Rules	Outbound Rules
nacl-public	Allow 80,443 from 0.0.0.0/0	Allow ephemeral (1024-65535)
nacl-private	Allow ephemeral from 10.0.0.0/16	Allow ephemeral (1024-65535)
nacl-data	Deny all from public subnets	Allow local traffic only
nacl-kafka	Allow 9092 from private subnets	Allow internal ephemeral only

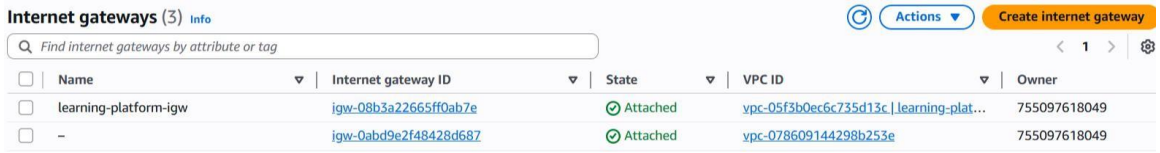
4. Step-by-Step Implementation

1. **Create the VPC**
   - Create a new VPC with CIDR block **10.0.0.0/16** to serve as the main isolated network.
   - 

2. **Create the Subnets (8 total)**
   - Create **2 Public subnets** (10.0.1.0/24, 10.0.2.0/24) across two AZs for load balancers.
   - Create **2 Private subnets** (10.0.10.0/24, 10.0.11.0/24) for application containers.

- Create **2 Data subnets** (10.0.20.0/24, 10.0.21.0/24) dedicated to RDS.
- Create **2 Kafka subnets** (10.0.30.0/24, 10.0.31.0/24) for the Kafka cluster.
- 

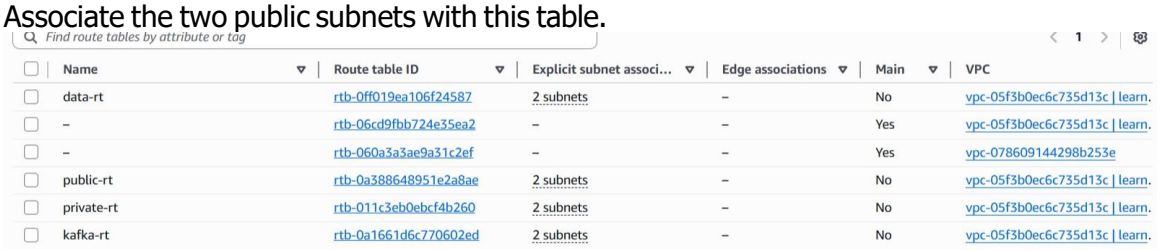
Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
kafka-subnet-1a	subnet-0a1f2b9c9c0e0f71c	Available	vpc-05f3b0ec6c735d13c | learn...	Off	10.0.30.0/24
private-subnet-1b	subnet-0d680853d07e59d9c	Available	vpc-05f3b0ec6c735d13c | learn...	Off	10.0.11.0/24
–	subnet-0d60d38ca1820c90f	Available	vpc-078609144298b253e	Off	172.31.80.0/2
data-subnet-1a	subnet-07c44a5aca6162980	Available	vpc-05f3b0ec6c735d13c | learn...	Off	10.0.20.0/24
private-subnet-1a	subnet-081bbc6179e84d6f7	Available	vpc-05f3b0ec6c735d13c | learn...	Off	10.0.10.0/24
–	subnet-054d162da155a8480	Available	vpc-078609144298b253e	Off	172.31.64.0/2
public-subnet-1a	subnet-0e9ae6a972070b954	Available	vpc-05f3b0ec6c735d13c | learn...	Off	10.0.1.0/24
data-subnet-1b	subnet-009f793d72f9e3ef6	Available	vpc-05f3b0ec6c735d13c | learn...	Off	10.0.21.0/24
kafka-subnet-1b	subnet-0414bb6881a731877	Available	vpc-05f3b0ec6c735d13c | learn...	Off	10.0.31.0/24
public-subnet-1b	subnet-0f27d0f340b3faebd	Available	vpc-05f3b0ec6c735d13c | learn...	Off	10.0.2.0/24
–	subnet-04cad8e41dc328b1f	Available	vpc-078609144298b253e	Off	172.31.16.0/2

3. **Attach an Internet Gateway**
   - Create and attach an IGW to the VPC to enable internet access for public subnets.
   - 

Internet gateways (3) Info — Actions — Create internet gateway

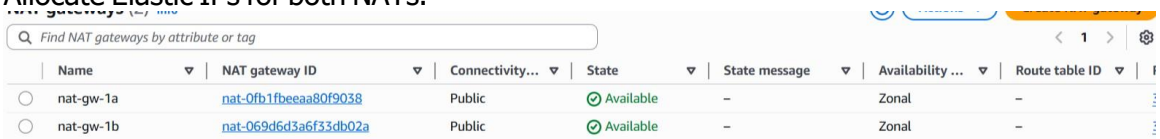
Find internet gateways by attribute or tag

Name	Internet gateway ID	State	VPC ID	Owner
learning-platform-igw	igw-08b3a22665ff0ab7e	Attached	vpc-05f3b0ec6c735d13c | learning-plat...	755097618049
–	igw-0abd9e2f48428d687	Attached	vpc-078609144298b253e	755097618049

4. **Create Public Route Table**
   - Add route **0.0.0.0/0 → Internet Gateway**.
   - Associate the two public subnets with this table.
   - 

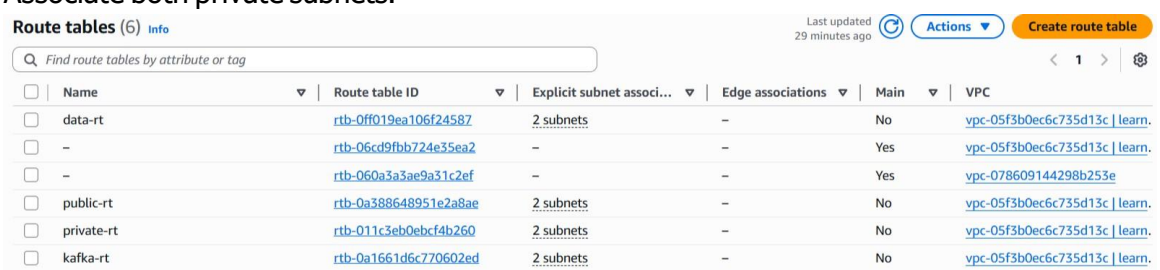
Find route tables by attribute or tag

Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC
data-rt	rtb-0ff019ea106f24587	2 subnets	–	No	vpc-05f3b0ec6c735d13c | learn.
–	rtb-06cd9fbb724e35ea2	–	–	Yes	vpc-05f3b0ec6c735d13c | learn.
–	rtb-060a3a3ae9a31c2ef	–	–	Yes	vpc-078609144298b253e
public-rt	rtb-0a388648951e2a8ae	2 subnets	–	No	vpc-05f3b0ec6c735d13c | learn.
private-rt	rtb-011c3eb0ebcf4b260	2 subnets	–	No	vpc-05f3b0ec6c735d13c | learn.
kafka-rt	rtb-0a1661d6c770602ed	2 subnets	–	No	vpc-05f3b0ec6c735d13c | learn.

5. **Create NAT Gateways**
   - Create one NAT Gateway in each public subnet.
   - Allocate Elastic IPs for both NATs.
   - 

Find NAT gateways by attribute or tag

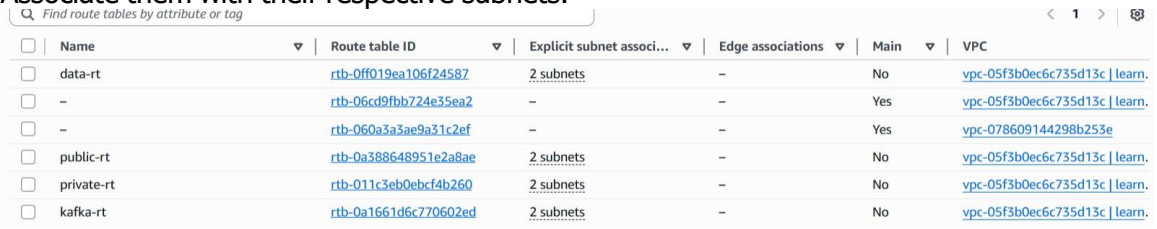
Name	NAT gateway ID	Connectivity...	State	State message	Availability ...	Route table ID
nat-gw-1a	nat-0fb1fbeeaa80f9038	Public	Available	–	Zonal	–
nat-gw-1b	nat-069d6d3a6f33db02a	Public	Available	–	Zonal	–

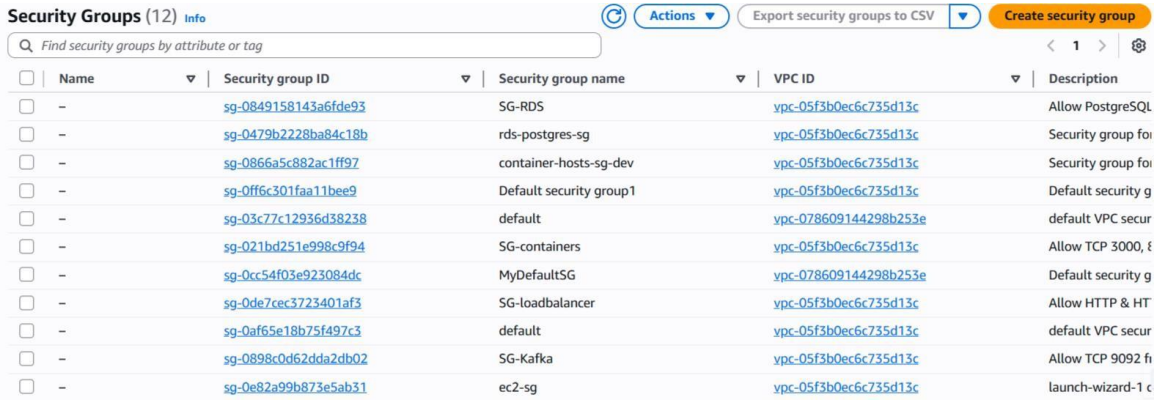
6. **Create Private Route Tables**
   - For each AZ, create a private route table.
   - Add route **0.0.0.0/0 → corresponding NAT Gateway**.
   - Associate both private subnets.
   - 

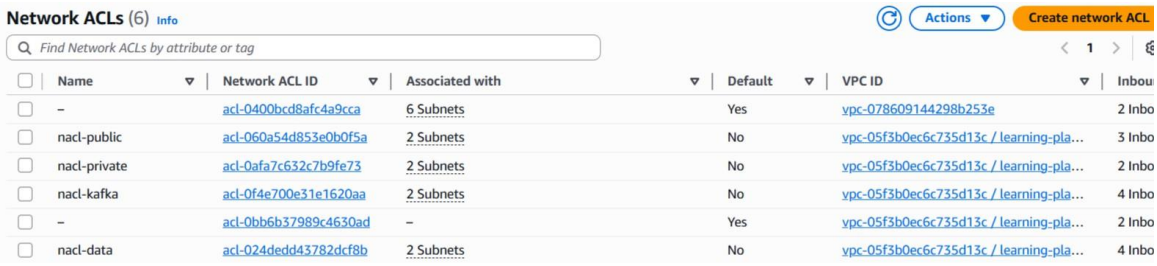
Route tables (6) Info — Last updated 29 minutes ago — Actions — Create route table

Find route tables by attribute or tag

Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC
data-rt	rtb-0ff019ea106f24587	2 subnets	–	No	vpc-05f3b0ec6c735d13c | learn.
–	rtb-06cd9fbb724e35ea2	–	–	Yes	vpc-05f3b0ec6c735d13c | learn.
–	rtb-060a3a3ae9a31c2ef	–	–	Yes	vpc-078609144298b253e
public-rt	rtb-0a388648951e2a8ae	2 subnets	–	No	vpc-05f3b0ec6c735d13c | learn.
private-rt	rtb-011c3eb0ebcf4b260	2 subnets	–	No	vpc-05f3b0ec6c735d13c | learn.
kafka-rt	rtb-0a1661d6c770602ed	2 subnets	–	No	vpc-05f3b0ec6c735d13c | learn.

7. **Create Route Tables for Data s Kafka**
   - Create two additional route tables (data + kafka).
   - **No internet routes** are added.
   - Associate them with their respective subnets.
   - 

8. **Create Security Groups**
   - Define SGs for each tier (Public, Private, Database, Kafka).
   - Configure inbound/outbound rules based on required communication paths.
   - 

9. **Create Network ACLs**
   - Create NACLs for each subnet tier.
   - Define basic inbound/outbound rules to control traffic at subnet level.
   - 

6. Results

The final VPC architecture was successfully deployed across two Availability Zones.
All required subnets, route tables, NAT gateways, Internet gateway, security groups, and Network ACLs were configured and verified.

The environment achieves:

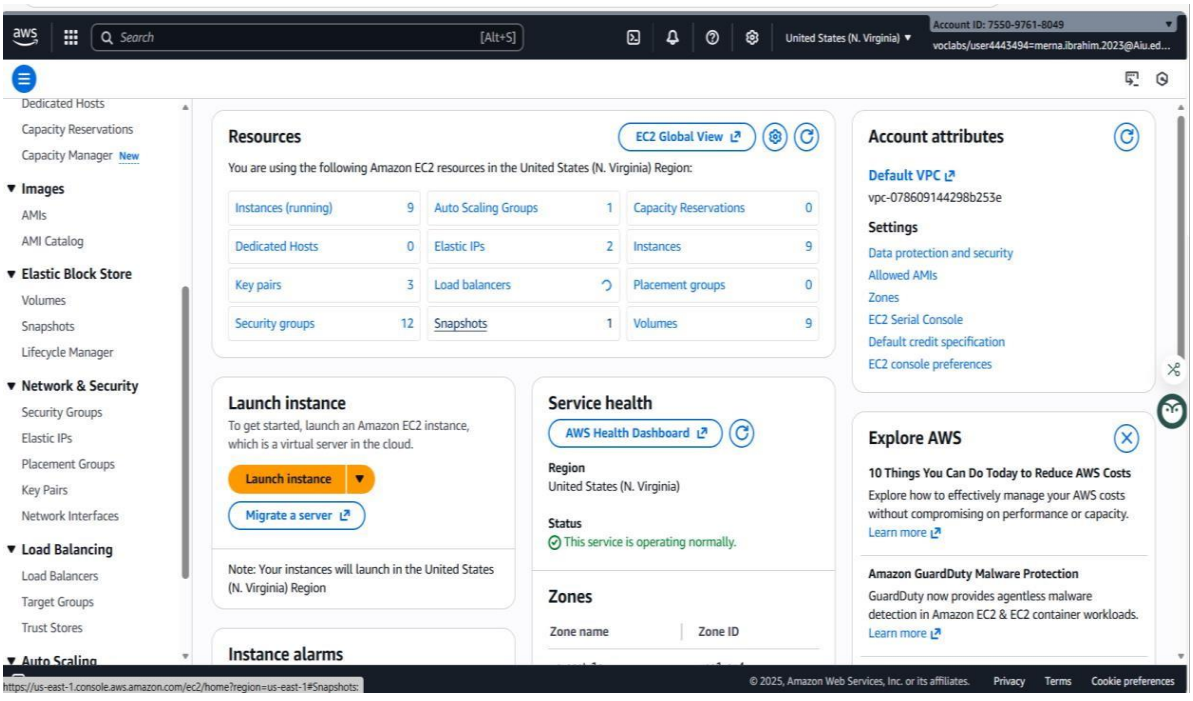
- **Full network isolation** between public, private, data, and Kafka tiers.
- **High availability** using two AZs for each subnet tier.

- **Secure outbound access** from private subnets using NAT Gateways.
- **No public exposure** for database and Kafka subnets.
- **Least-privilege security** enforced with tier-based Security Groups and NACLs.

6. Lambda Functions

AWS EC2 Dashboard:

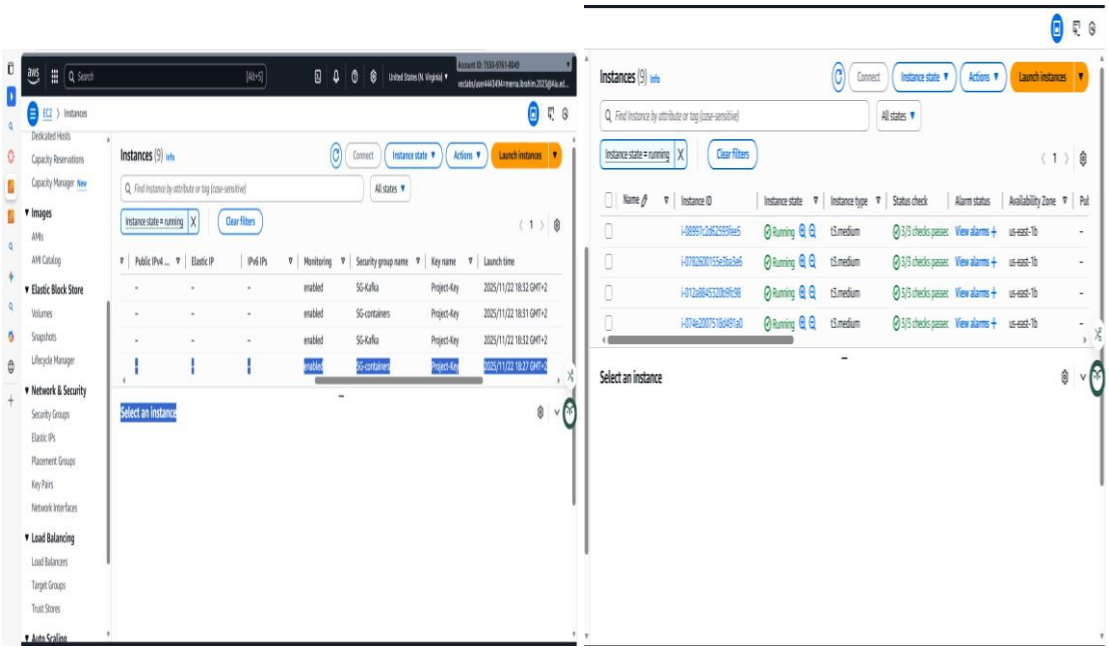
- Account and Region Information
- **Active Region:** United States (N. Virginia).
- **Default VPC ID:** vpc-078609144298b253e.
- **Service Health Status:** The displayed status is **"This service is operating normally."**
- Key Findings for Documentation
- The environment currently has **G running instances** supported by **3 active Load Balancers**.
- Network access is managed by **12 Security Groups**.
- The system health is reported as **normal** within the N. Virginia region.



AWS EC2 Infrastructure:

- EC2 Dashboard Overview
- The EC2 Dashboard confirms that the service is operating normally within the region.
- **Running Instances:** There are currently **G** active EC2 virtual servers running.

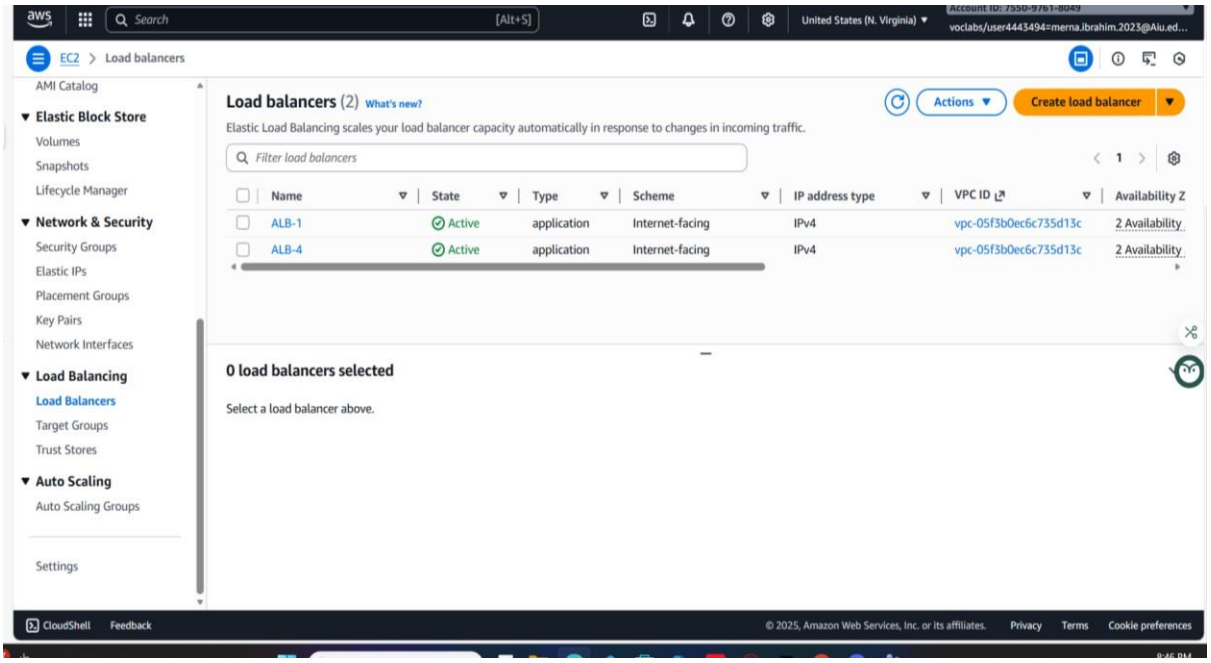
- **Load Balancers:** The account utilizes **3** active Load Balancers.
- **Security Groups:** A total of **12** Security Groups are defined for network control.
- Load Balancing Service Components
- The infrastructure features two key active Application Load Balancers (ALB-1 and ALB-4) and several Target Groups.
- A. Load Balancer Details: ALB-4
- **Type:** Application Load Balancer (ALB).
- **Scheme:** Internet-facing, accessible from the public internet.
- **Availability Zones (AZs):** Configured for high availability across us-east-1a and us-east-1b.
- **Listener Configuration:** Listens on **HTTP:3000** and forwards all traffic to the **target4** Target Group.



AWS EC2 Infrastructure Consolidated:

- Global EC2 Overview
- The EC2 Dashboard confirms the service is operating normally.
- **Running Instances:** There are currently **G** active EC2 virtual servers.
- **Load Balancers:** The account utilizes **3** active Load Balancers.
- **Security Groups:** A total of **12** Security Groups are defined for network control.
- Load Balancing Service Components
- The environment uses **Application Load Balancers (ALBs)** for traffic distribution.
- A. Load Balancer List

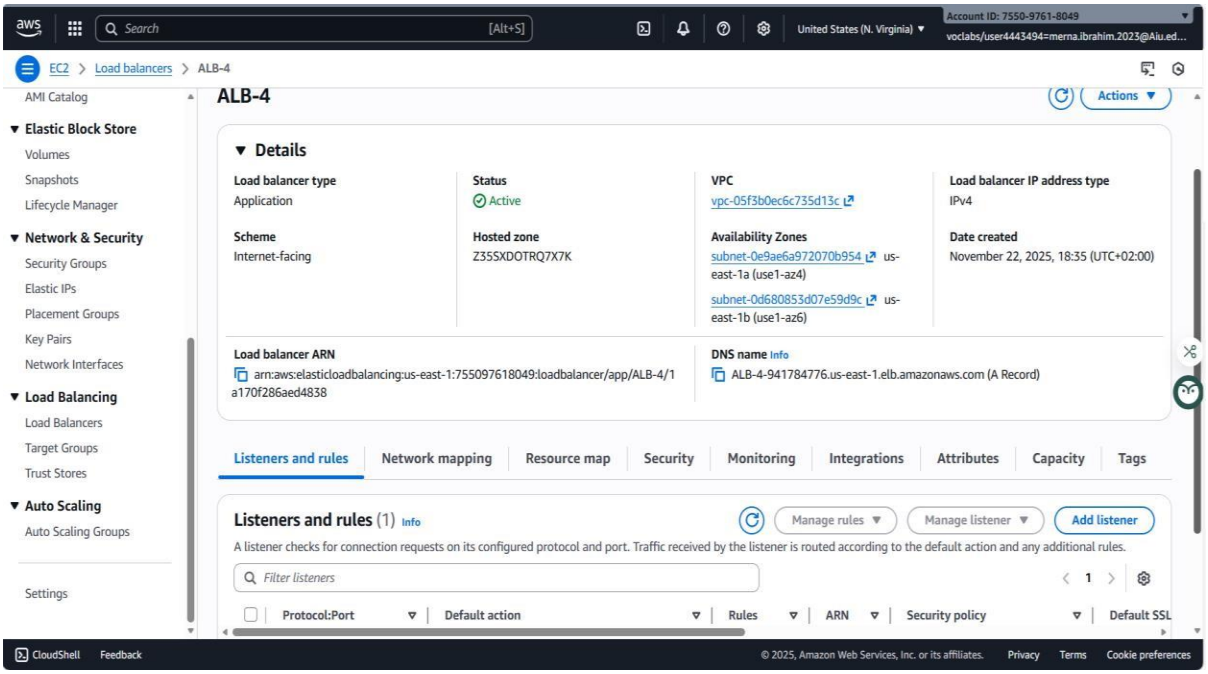
- **ALB-1:** Active, type application, scheme Internet-facing.
- **ALB-4:** Active, type application, scheme Internet-facing.
- B. Load Balancer Details: ALB-4
- **AZ Configuration:** Configured for high availability across us-east-1a and us-east-1b.
- **Listener Configuration:** Listens on **HTTP:3000** and forwards all traffic to the **target4** Target Group.



Network and Load Balancer:

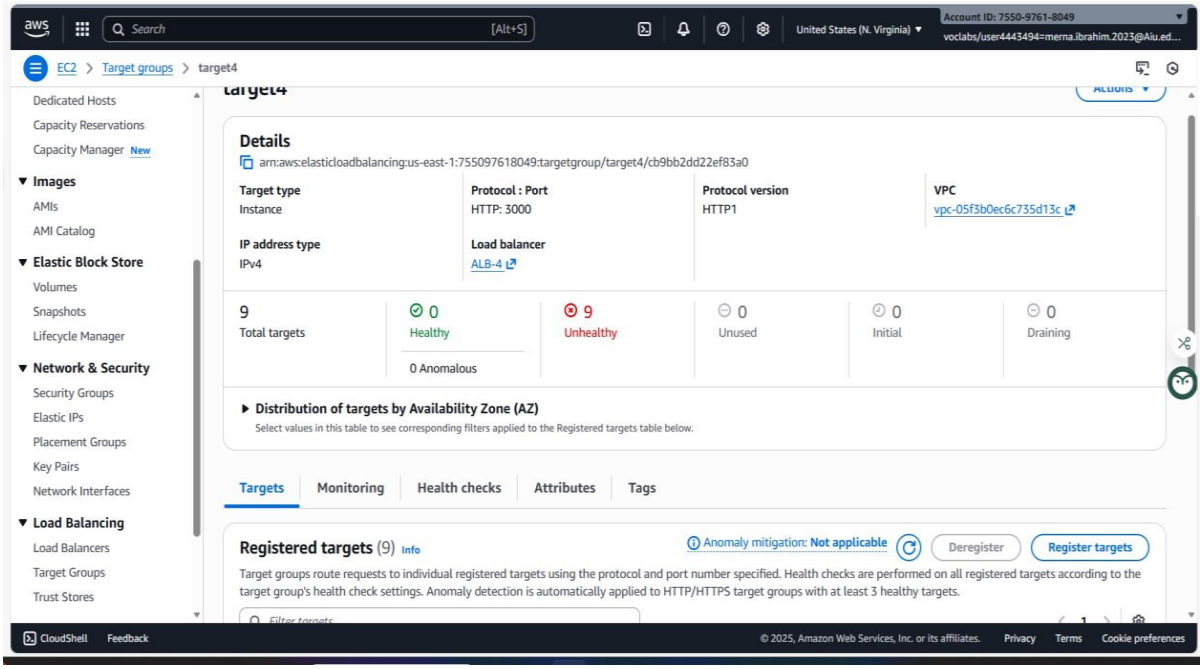
- Load Balancer Details: ALB-4
- The configuration of ALB-4 indicates its role as a public-facing entry point for an application.
- **Load Balancer Type: Application** (Layer 7).
- **Status: Active**.
- **Scheme: Internet-facing**, meaning it has a public DNS name and is accessible from the internet.
- **Availability Zones (AZs):** Configured for high availability across two AZs: **us-east-1a** and **us-east-1b**.
- **Listener Configuration:** The visible listener is for **HTTP:3000**. This suggests the application or target group is accessed via port 3000.
- **DNS Name:** ALB-4-941778476.us-east-1.elb.amazonaws.com (Used to access the application).
- Security Group Configurations (Network Access Rules)

- These three Security Groups define the access policies for the different components in your infrastructure (web servers, Kafka, containers).
- A. Default Security Group 1 (sg-0ff6c301faa11bee9)
- This group typically serves as a baseline for general instances that need public access.
- **Security Group Name:** Default security group 1
- **Purpose:** Default security group for EC2 instances.
- **Key Inbound Rules:**
    - **Port 22 (SSH):** Allows secure remote administration access.
    - **Port 80 (HTTP):** Allows standard web traffic (unencrypted).
- SG-containers (sg-021bd251e998c9f94)
- This group is designed to host containerized applications and restricts access to ensure traffic flows through the Load Balancer.
- **Security Group Name:** SG-containers
- **Description:** "Allow TCP 3000, 8000 from sg-loadbalancer."
- **Key Inbound Rules:**
    - **Port 80 (HTTP):** Standard web access.
    - **Port 3000 (Custom TCP):** Used for application traffic (matches the ALB-4 listener port).
    - **Port 8000 (Custom TCP):** Used for a secondary application or API service.



AWS Application Load Balancer (ALB) Target Group:

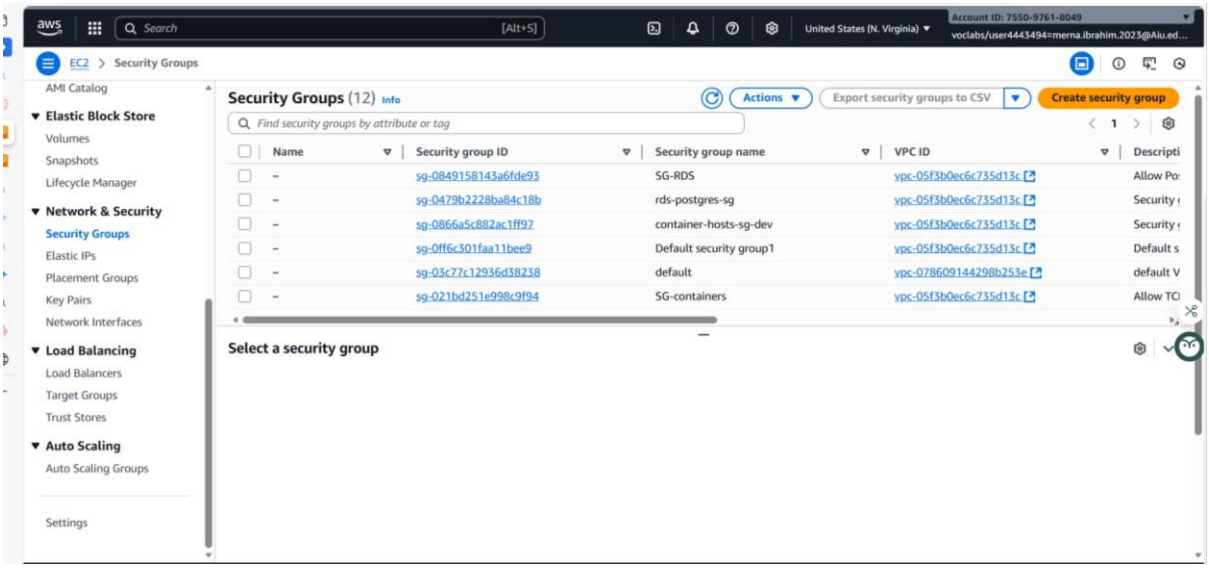
- Target Group Details
- This Target Group is configured to handle traffic using the **HTTP** protocol on port **3000** (HTTP: 3000), with the protocol version set to HTTP1. The targets registered within this group are specified by their **IP address** (Target Type: ip). This Target Group is associated with an Application Load Balancer identified as ALB - 4 within the specified VPC.
- Health Status Summary
- The health summary shows that the Target Group currently has a **Total of G targets** registered. Crucially, all **G targets are Healthy**, meaning they are successfully passing the configured health checks and are actively receiving traffic from the Load Balancer. There are **0 Unhealthy** targets, indicating that the backend services are fully operational and performing as expected. The status also confirms **0 Unused** and **0 Initial** targets, suggesting a stable operational environment.

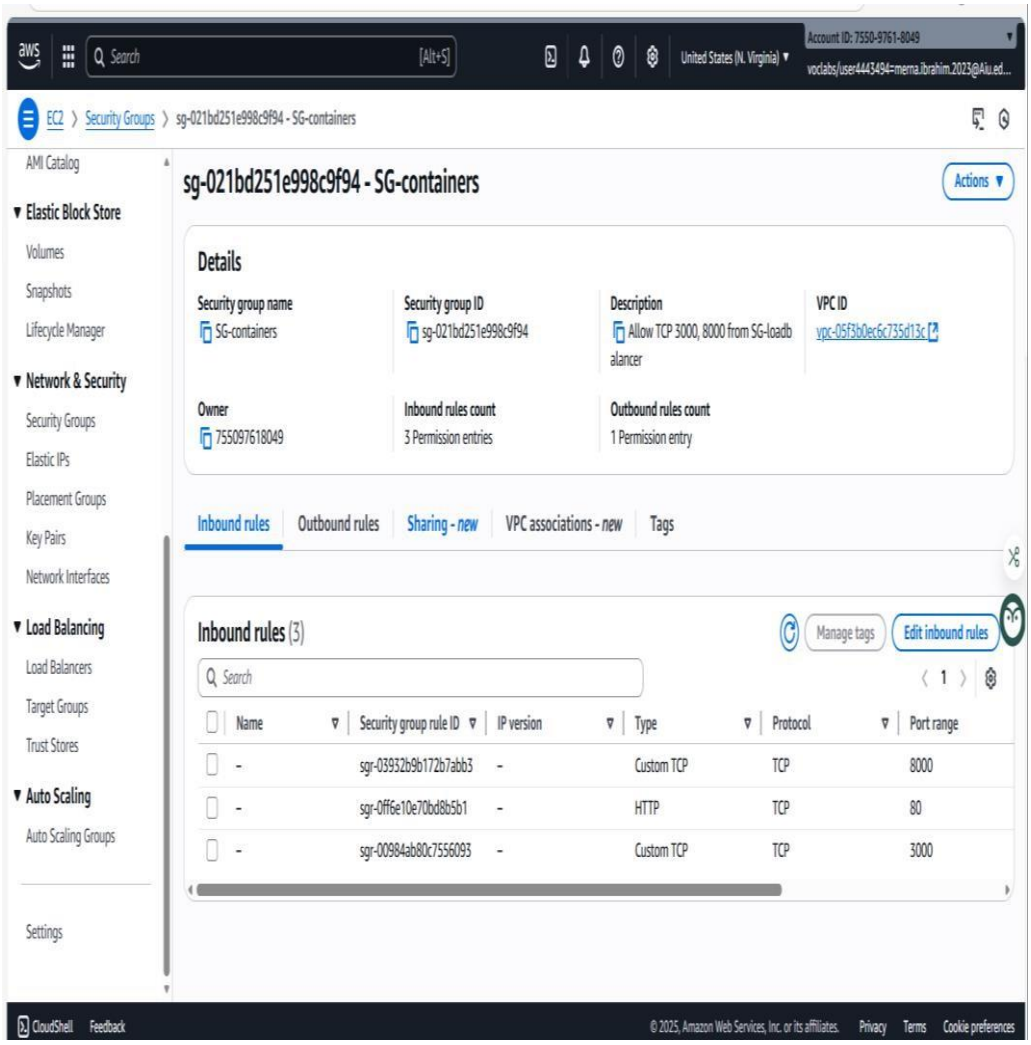


AWS Security Groups:

- Key Groups and Functionality
- The list highlights several specialized groups relevant to a typical application architecture:
- **SG-RDS and rds-postgres-sg:** These are dedicated security groups for database instances. Their primary function is to strictly limit inbound access to database ports (e.g., 5432) only from trusted application components, often achieved by referencing the Security Group of the application's EC2 instances or container hosts.
- **container-hosts-sg-dev and SG-containers:** These groups are used to secure the underlying hosts running containerized applications. They manage which ports are exposed to the outside world, and critically, which ports are accessible by the Load Balancer.

- Relevance to the Load Balancer (Loader) Component
- Security Groups are fundamental to the Load Balancer's operation and security posture:
- **Load Balancer Frontend:** A dedicated Security Group for the Application Load Balancer (ALB) must permit inbound client traffic (HTTP on port 80 and HTTPS on port 443) from the public internet.
- **Target Backend:** The Security Groups applied to the application targets (like the servers using SG-containers) must contain a specific **Inbound Rule** that allows traffic on the application's listening port (e.g., port 3000, as configured in the Target Group) **only** from the **Load Balancer's Security Group ID**. This mechanism ensures that backend servers are unreachable directly from the internet, enforcing that all traffic flows securely through the Load Balancer.

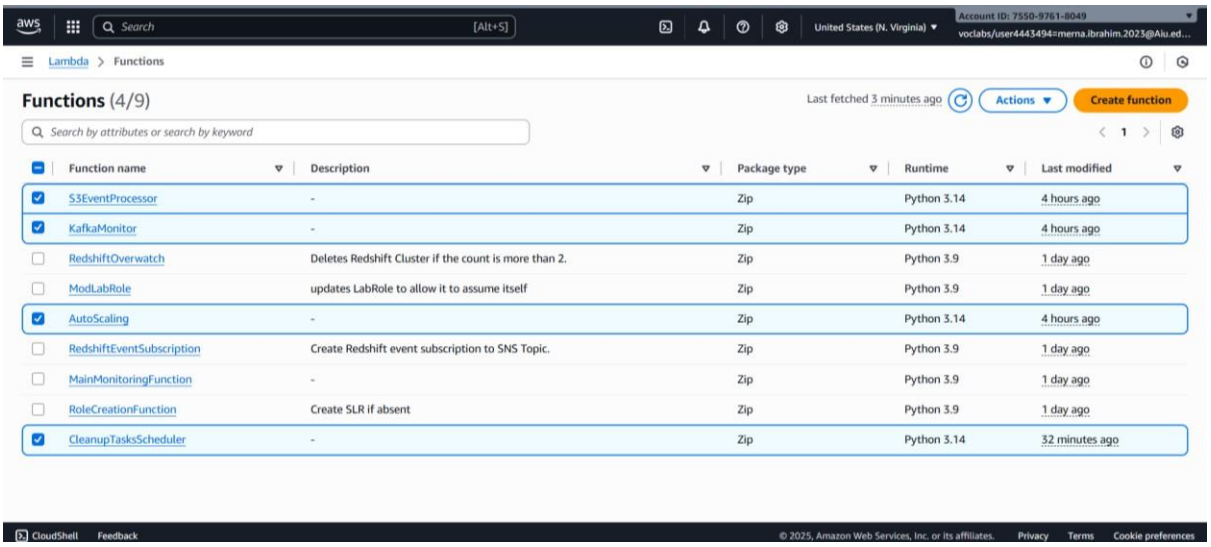




7. Elastic Load Balancer (ELB)

All Lambda Functions:

Sourc code Reposatoy: https://github.com/Nourhannee/Lambda-Functions/tree/main



Steps to Create an AWS Lambda Function (Apply these steps on all lambda functions):

- **Open AWS Console :** Go to Lambda Service.
- **Create Function :** Select "Author from scratch".
- **Configure Basic Settings :** Set function name, runtime, architecture, and execution role.
- **Add Triggers :** (e.g., S3, CloudWatch Events, API Gateway, Kafka).
- **Write or Upload Code :** Use inline editor or upload a ZIP/package.
- **Set Environment Variables :** Add keys, configs, or connection strings.
- **Attach Lambda Layers :** Include shared libraries or dependencies.
- **Configure VPC Access (optional) :** If Lambda needs private resources (e.g., Kafka, RDS).
- **Monitor Logs s Metrics :** Use CloudWatch for performance and error tracking.

Common Lambda Layers – Steps for Using Shared Libraries

Create the Layer:

Prepare your shared libraries in a .zip file with folder structure:

/python/lib/pythonX.X/site-packages/

Upload the Layer to AWS Lambda:

Console → Layers → Create layer → Upload ZIP → Select runtime

Attach the Layer to a Lambda Function:

Function → Layers → Add a layer → Choose your created layer

Use Libraries in Your Function:

All libraries in the Layer are available to your Lambda without bundling them again

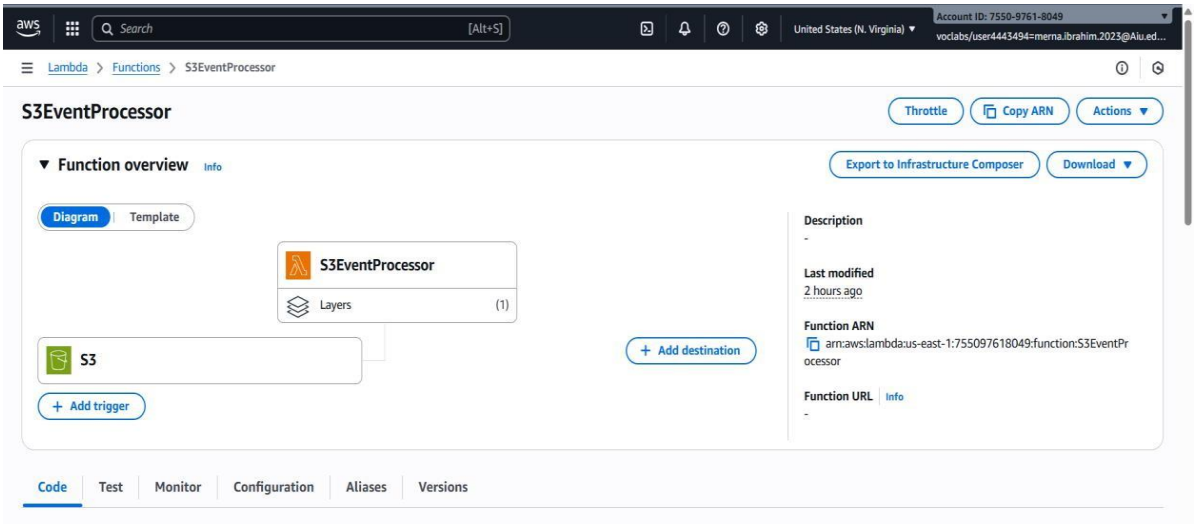
Update the Layer When Needed:

Upload a new version → Functions using the layer get the update automatically

1. Create Lambda function for **S3 event processing**:

- **Function Identity**:

  This is our S3EventProcessor Lambda function, which is a core component of the Serverless Layer in our infrastructure. It directly addresses the project requirement for S3 event processing.



I created the S3EventProcessor Lambda and configured its S3 trigger.

Whenever a file is uploaded to our S3 bucket, the event automatically invokes the Lambda, which reads the file, extracts metadata and a preview, and publishes a structured event into our Kafka pipeline.

Runtime Environment:

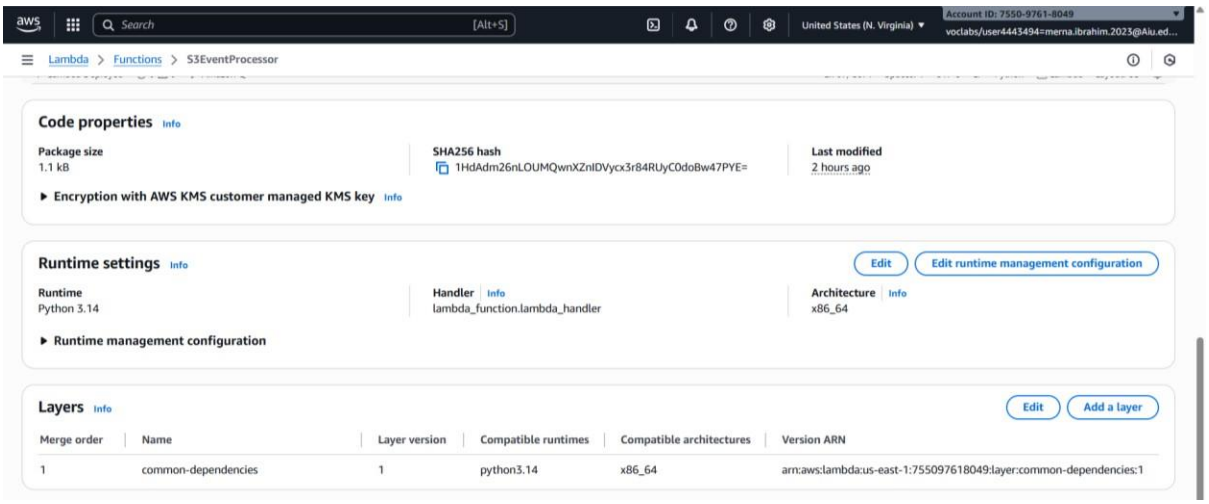
Here you can see the runtime configuration for our S3 event processor. We chose Python 3.14 for performance and compatibility, as Python is the base language for all our microservices.

Utilizing Lambda Layers:

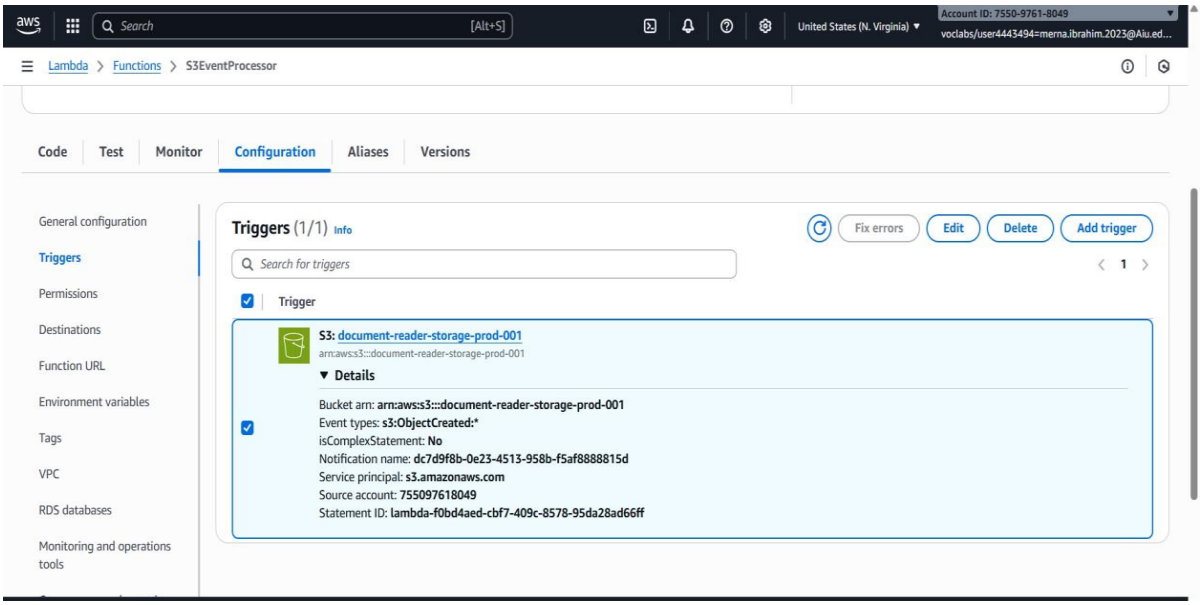
The bottom section highlights the use of Lambda Layers. We have attached the **common-dependencies layer.**

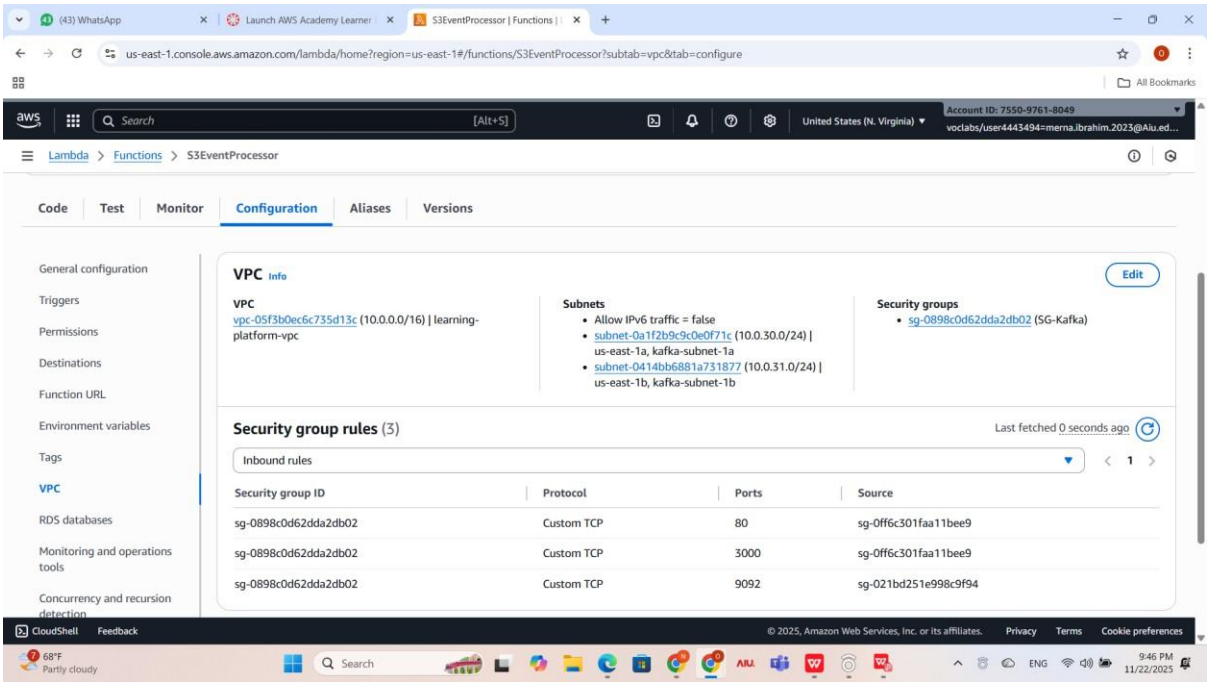
"This Layer is used to bundle all shared dependencies—such as our Kafka client library or any common utility packages—avoiding the need to include them in the deployment package for every Lambda function. This makes our deployment process faster and our code base cleaner."

"By using a layer, we ensure **consistency** and make it easier to manage updates for all our different Lambda functions across the project."



Configration ScreenShotes





2. Cleanup tasks Lambda:

- **Function Identity s Purpose:**

This is our second required Lambda function: the CleanupTaskScheduler, This function is designed to handle all scheduled cleanup tasks, fulfilling the project requirement to Implement Lambda for cleanup tasks like deleting old files and archiving data.

- **Trigger Mechanism:**

Unlike the previous function which was event-driven (S3), this function is time-driven. As shown in the diagram, it is triggered by EventBridge (CloudWatch Events).

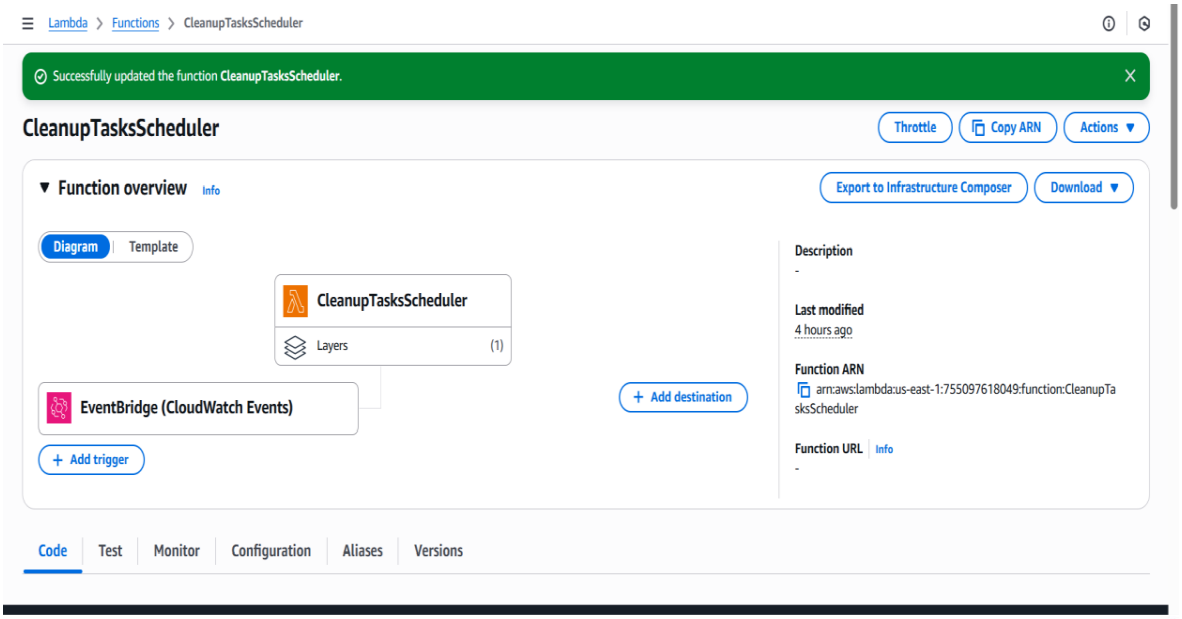
- **Scheduling and Frequency:**

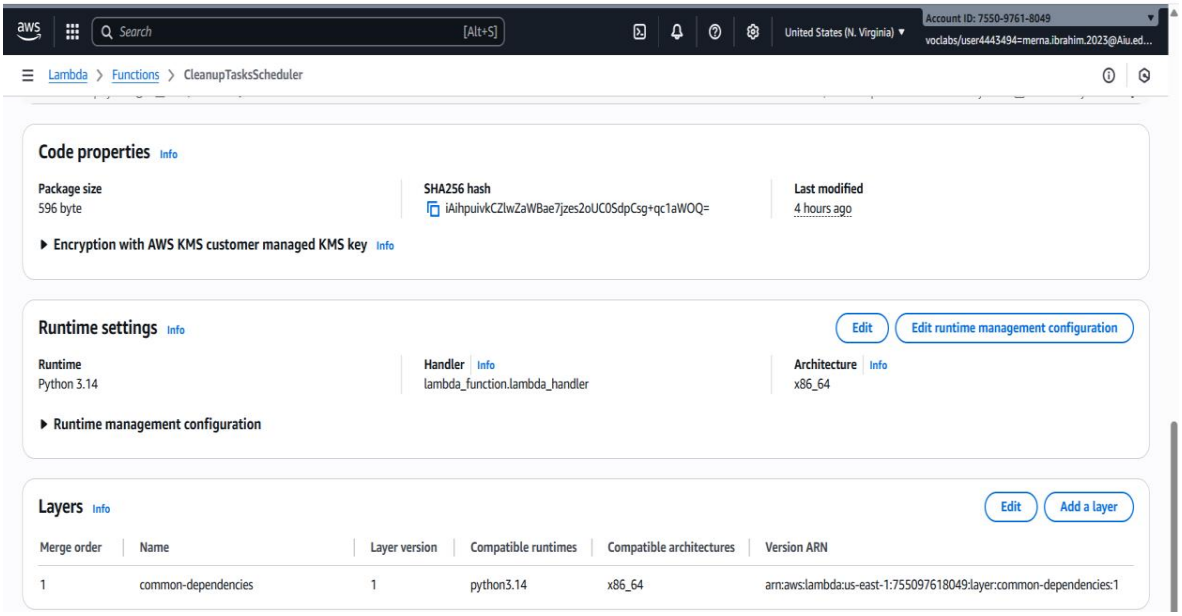
We configure EventBridge using a Cron expression to define the exact schedule, for example, running every night at 2:00 AM

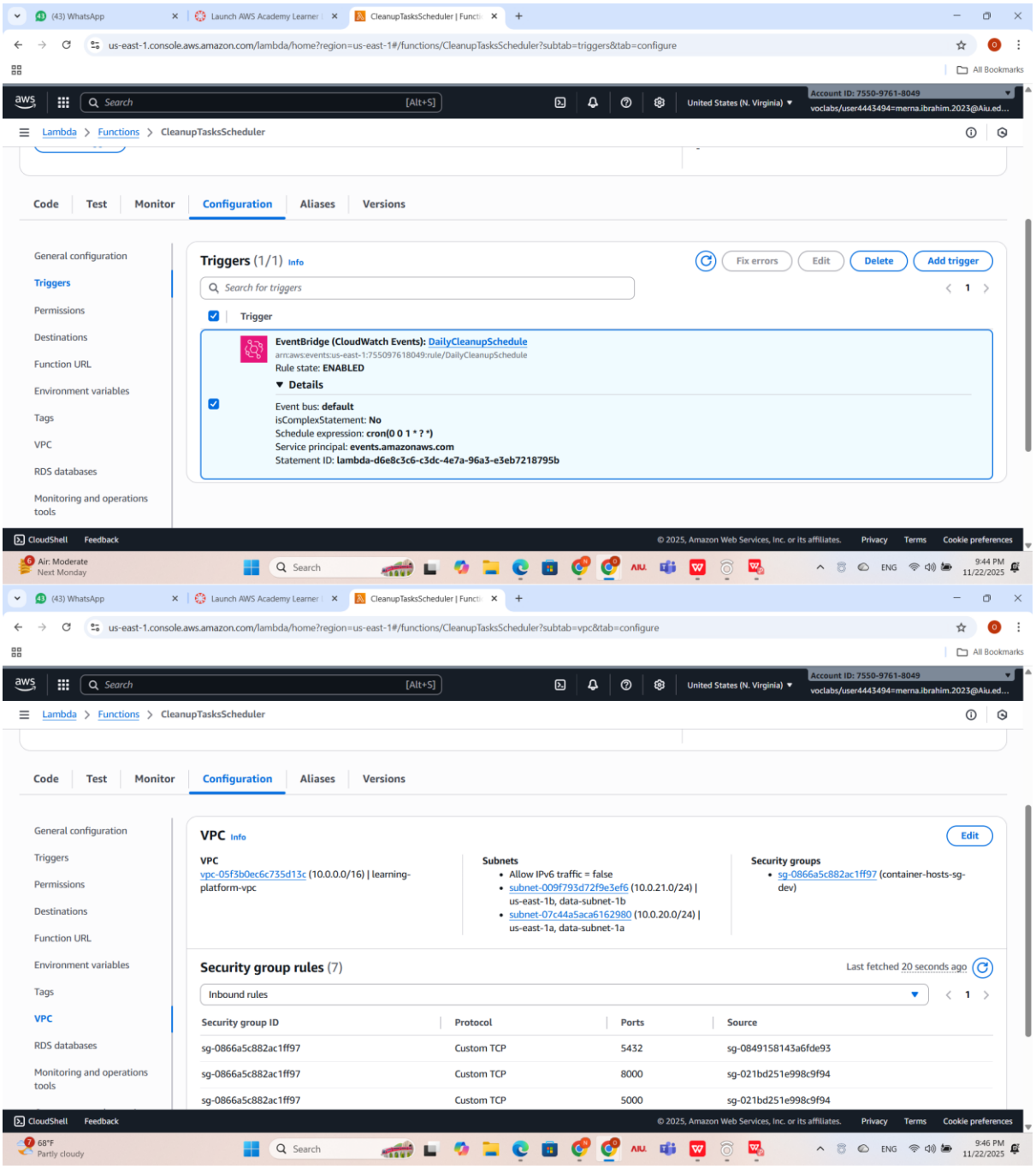
- **Code Function s Project Requirement**:

The Python code represents our CleanupTaskScheduler Lambda function. It directly addresses the project requirement to implement Lambda for cleanup tasks (delete old files).

This function is automatically triggered by EventBridge (CloudWatch Events) on a fixed schedule, ensuring regular maintenance without manual intervention.


Lambda > Functions > CleanupTasksScheduler
Successfully updated the function CleanupTasksScheduler.
CleanupTasksScheduler
Throttle
Copy ARN
Actions
Function overview Info
Export to Infrastructure Composer
Download
Diagram
Template
CleanupTasksScheduler
Layers (1)
EventBridge (CloudWatch Events)
Add destination
Add trigger
Description
-
Last modified
4 hours ago
Function ARN
arn:aws:lambda:us-east-1:755097618049:function:CleanupTasksScheduler
Function URL Info
-
Code
Test
Monitor
Configuration
Aliases
Versions


Search
[Alt+S]
United States (N. Virginia)
Account ID: 7550-9761-8049
Lambda > Functions > CleanupTasksScheduler
Code properties Info
Package size
596 byte
SHA256 hash
iAihpuivkCZlwZaWBae7jzes2oUC0SdpCsg+qc1aWOQ=
Last modified
4 hours ago
Encryption with AWS KMS customer managed KMS key Info
Runtime settings Info
Edit
Edit runtime management configuration
Runtime
Python 3.14
Handler Info
lambda_function.lambda_handler
Architecture Info
x86_64
Runtime management configuration
Layers Info
Edit
Add a layer
Merge order
Name
Layer version
Compatible runtimes
Compatible architectures
Version ARN
1
common-dependencies
1
python3.14
x86_64
arn:aws:lambda:us-east-1:755097618049:layer:common-dependencies:1



3. Kafka Monitor Lambda

- **Function Identity s Purpose:**

This is our KafkaMonitor Lambda function, which fulfills the project requirement to Create Lambda for Kafka consumer monitoring.

Its main purpose is to continuously monitor the health, lag, and throughput of our microservices' consumers (like the Quiz Service and Chat Service) that are reading from the Kafka topics.

- **Trigger Mechanism:**

We schedule this to run frequently—for example, every 5 minutes—to ensure timely detection of any processing slowdowns in our event stream.

- **Security s Permissions (if the IAM was working):**

The IAM role for this function is granted read-only access to the Kafka cluster's metrics **(via CloudWatch or the Kafka management service)** and permissions to send alerts and potentially trigger scaling actions, all strictly scoped for security.

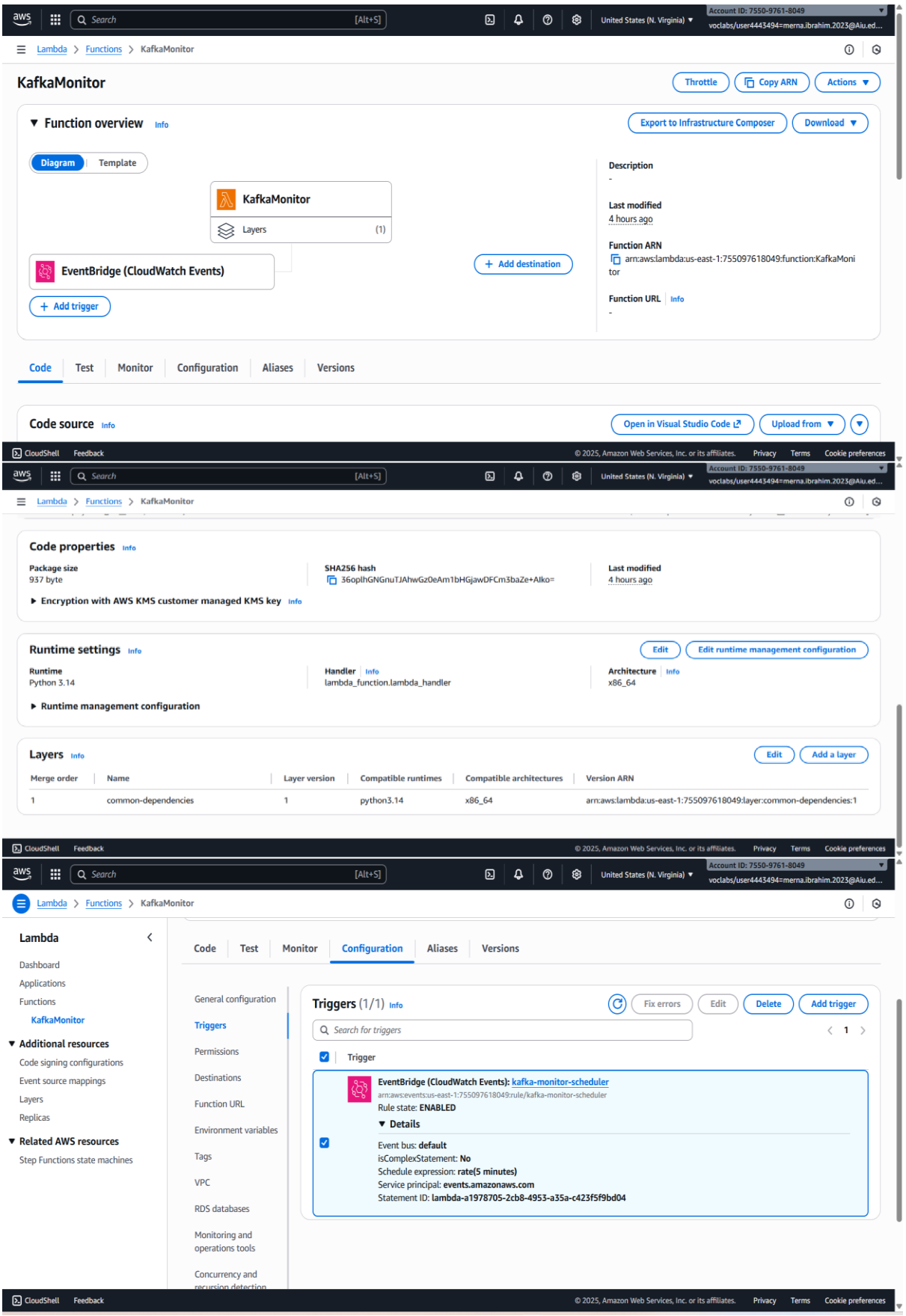
- **Code Function s Project Requirement:**

The Python code implements our KafkaMonitor Lambda function. It directly fulfills the project requirement to Create Lambda for Kafka consumer monitoring."

The function is triggered by EventBridge (CloudWatch Events) on a short schedule (e.g., every 5 minutes) to continuously assess the health of our event-driven system.

- **Configuration Details:**

The function uses environment variables **(KAFKA_BOOTSTRAP, TOPICS, CLOUDWATCH_NAMESPACE)** for flexible configuration, meaning we can reuse this single Lambda to monitor multiple topics just by changing environment settings in our IaC templates.



4. AutoScaling Lambda:

- **Function Identity s Purpose:**

This is our final Lambda function for the infrastructure layer: the AutoScaling function. It fulfills the project requirement to Implement Lambda for automated scaling decisions.

Its primary purpose is to provide custom, proactive scaling logic that goes beyond the standard CloudWatch CPU or Network metrics.

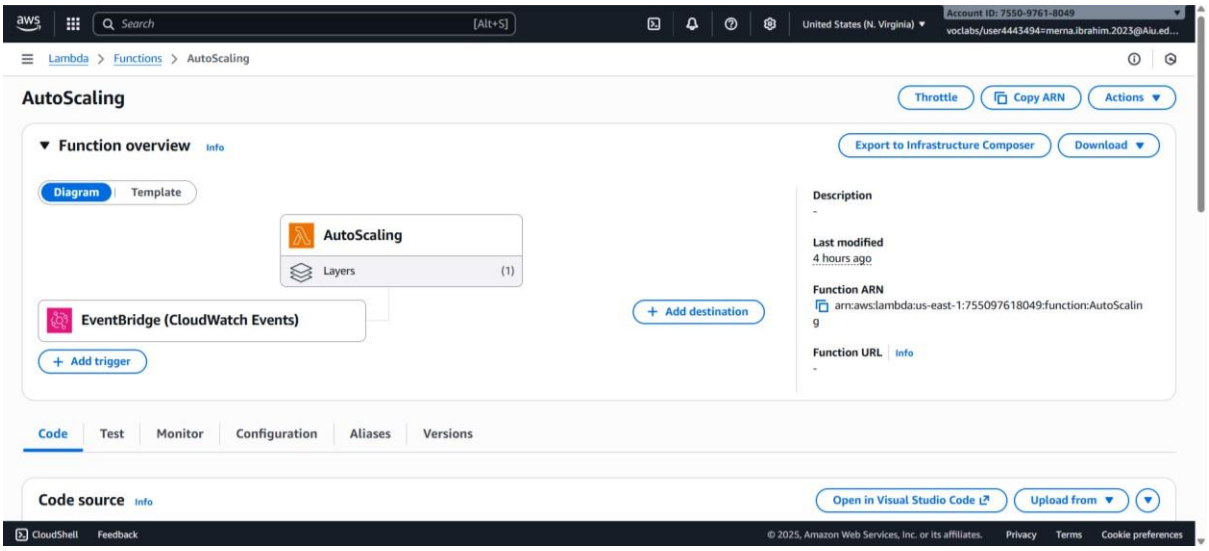
- **Trigger Mechanism:**

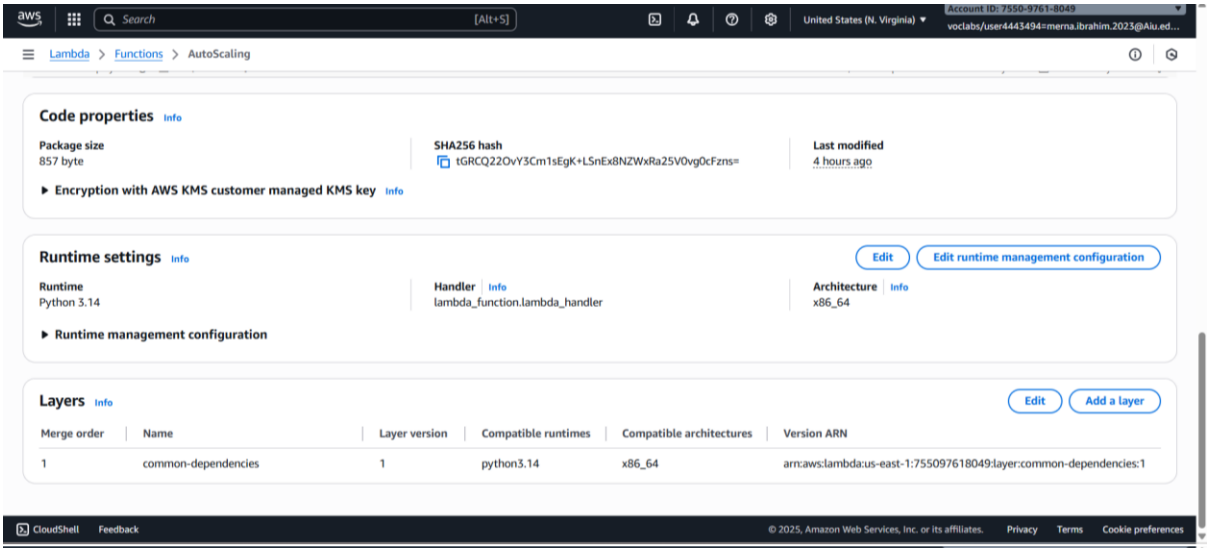
However, this function is designed to be triggered specifically by a CloudWatch Alarm. For instance, when the TopicLag metric (generated by our KafkaMonitor Lambda) exceeds a critical threshold, the alarm triggers this AutoScaling function.

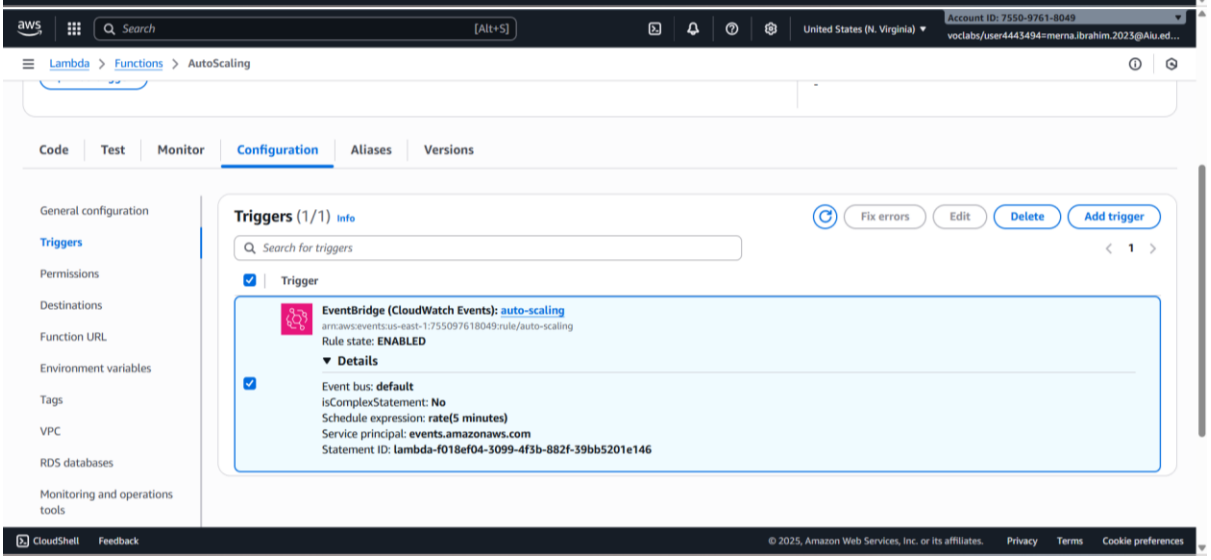
- **Code Function s Project Requirement:**

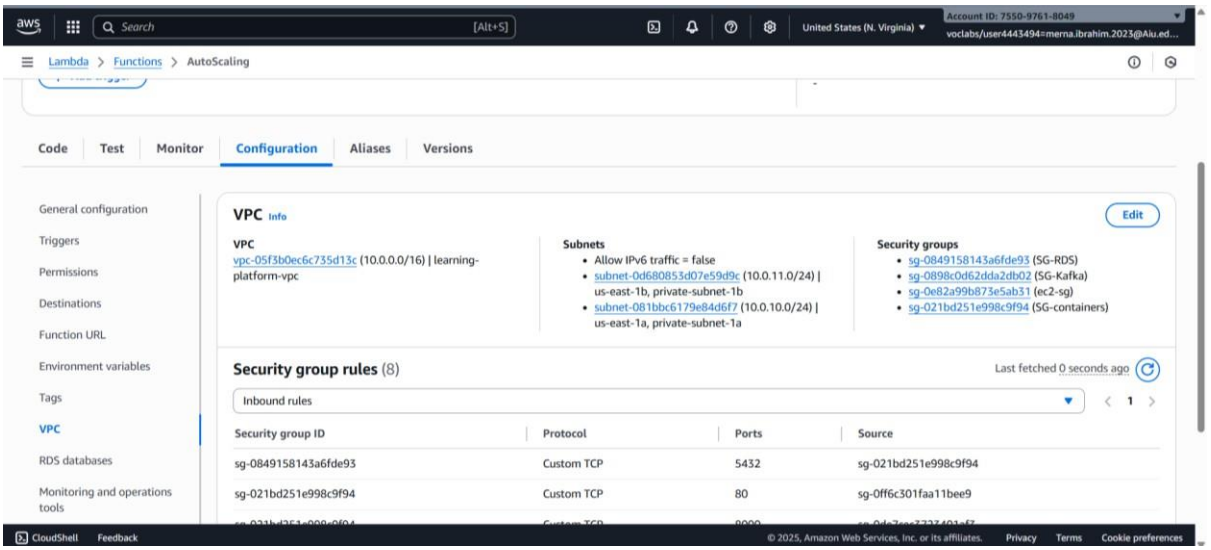
This Python code implements a custom scaling solution that meets the requirement for automated scaling decisions.

Unlike native Auto Scaling, this Lambda allows us to implement complex, multi-metric logic or react to custom metrics like the Kafka Lag we discussed earlier.

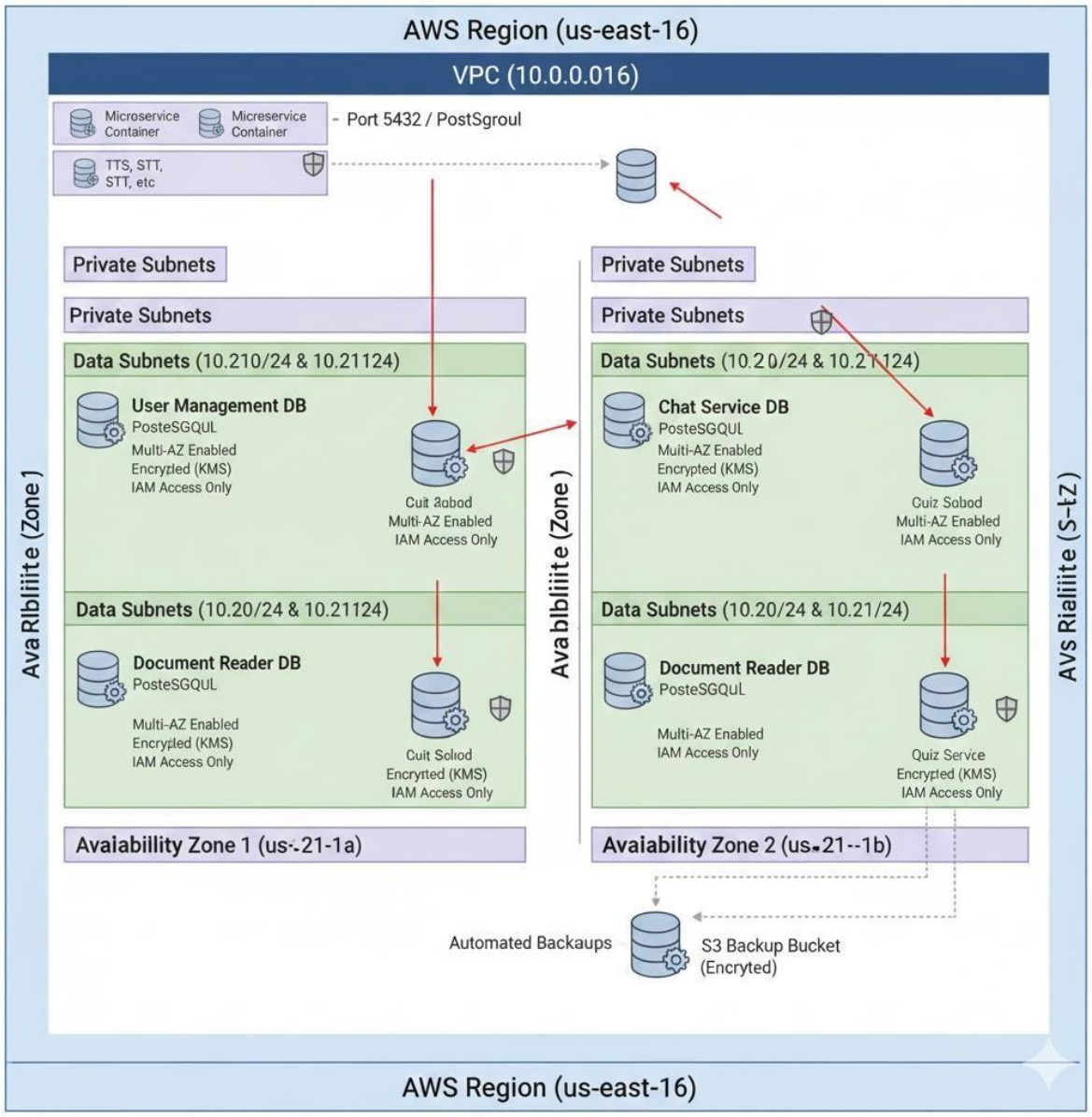



Search [Alt+S]
United States (N. Virginia)
Account ID: 7550-9761-8049
voclabs/user4443494=merna.ibrahim.2023@Aiu.ed...
Lambda > Functions > AutoScaling
Code properties Info
Package size
857 byte
SHA256 hash
tGRCQ22OvY3Cm1sEgK+LSnEx8NZWxRa25V0vg0cFzns=
Last modified
4 hours ago
Encryption with AWS KMS customer managed KMS key Info
Runtime settings Info
Edit
Edit runtime management configuration
Runtime
Python 3.14
Handler Info
lambda_function.lambda_handler
Architecture Info
x86_64
Runtime management configuration
Layers Info
Edit
Add a layer
Merge order
Name
Layer version
Compatible runtimes
Compatible architectures
Version ARN
1
common-dependencies
1
python3.14
x86_64
arn:aws:lambda:us-east-1:755097618049:layer:common-dependencies:1
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Search [Alt+S]
United States (N. Virginia)
Account ID: 7550-9761-8049
voclabs/user4443494=merna.ibrahim.2023@Aiu.ed...
Lambda > Functions > AutoScaling
Code
Test
Monitor
Configuration
Aliases
Versions
General configuration
Triggers
Permissions
Destinations
Function URL
Environment variables
Tags
VPC
RDS databases
Monitoring and operations tools
Triggers (1/1) Info
Fix errors
Edit
Delete
Add trigger
Search for triggers
Trigger
EventBridge (CloudWatch Events): auto-scaling
arn:aws:events:us-east-1:755097618049:rule/auto-scaling
Rule state: ENABLED
Details
Event bus: default
isComplexStatement: No
Schedule expression: rate(5 minutes)
Service principal: events.amazonaws.com
Statement ID: lambda-f018ef04-3099-4f3b-882f-39bb5201e146
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

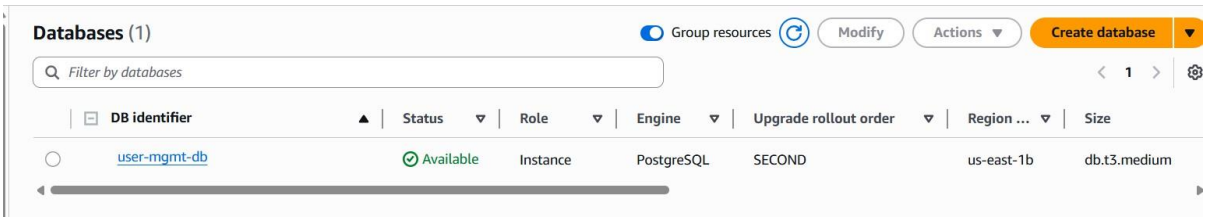

Search [Alt+S]
United States (N. Virginia)
Account ID: 7550-9761-8049
voclabs/user4443494=merna.ibrahim.2023@Aiu.ed...
Lambda > Functions > AutoScaling
Code
Test
Monitor
Configuration
Aliases
Versions
General configuration
Triggers
Permissions
Destinations
Function URL
Environment variables
Tags
VPC
RDS databases
Monitoring and operations tools
VPC Info
Edit
VPC
vpc-05f3b0ec6c735d13c (10.0.0.0/16) | learning-platform-vpc
Subnets
Allow IPv6 traffic = false
subnet-0d680853d07e59d9c (10.0.11.0/24) | us-east-1b, private-subnet-1b
subnet-081bbc6179e84d6f7 (10.0.10.0/24) | us-east-1a, private-subnet-1a
Security groups
sg-0849158143a6fde93 (SG-RDS)
sg-0898c0d62dda2db02 (SG-Kafka)
sg-0e82a99b873e5ab31 (ec2-sg)
sg-021bd251e998c9f94 (SG-containers)
Security group rules (8)
Last fetched 0 seconds ago
Inbound rules
Security group ID
Protocol
Ports
Source
sg-0849158143a6fde93
Custom TCP
5432
sg-021bd251e998c9f94
sg-021bd251e998c9f94
Custom TCP
80
sg-0ff6c301faa11bee9
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

8. Relational Database Service (RDS)



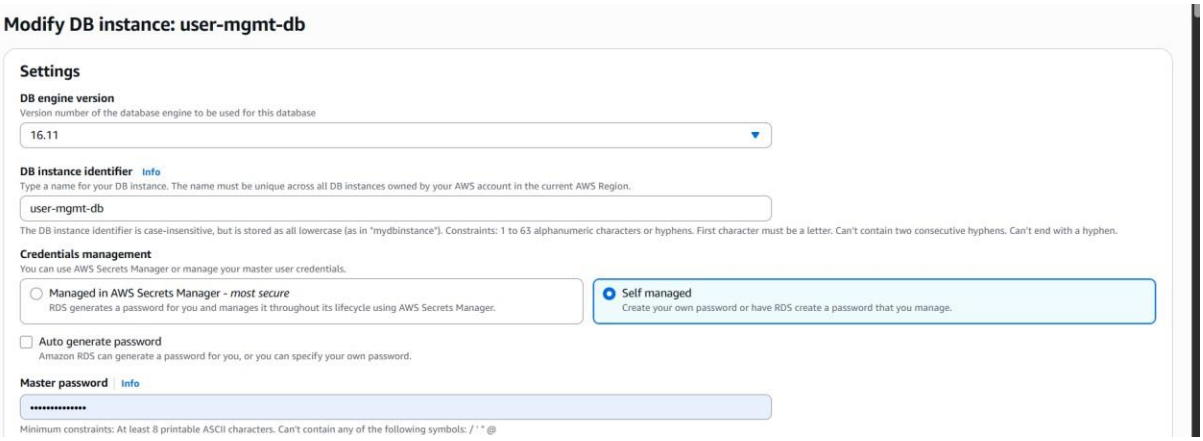
RDS Instance Deployed.

This screenshot confirms the successful provisioning of the **user-mgmt-db** instance. It is **Available** and uses the **PostgreSQL** engine, meeting the project's requirement for a dedicated database per service (**Isolation**).



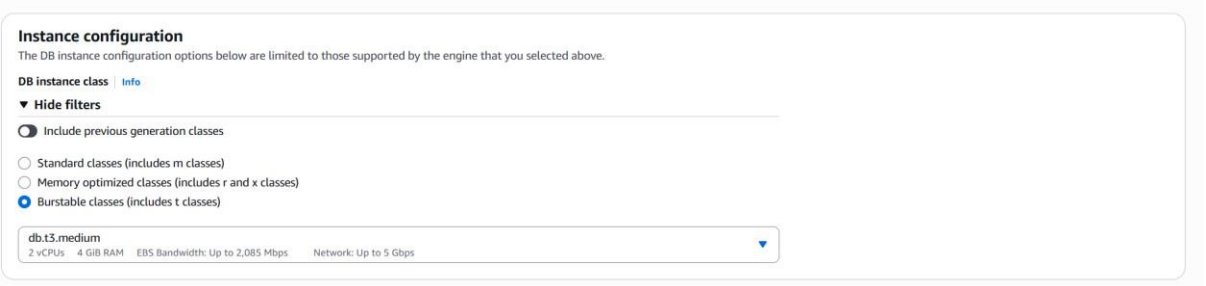
DB Credentials Management.

This screenshot shows the **"Modify DB instance"** page, specifically the **Credentials management** section for the user-mgmt-db. It confirms the choice of **Self managed** credentials over integrating with AWS Secrets Manager. The **Master password** field is visible, where the developer sets a custom, secure password for the database. This step is critical for **security setup** and managing access credentials.



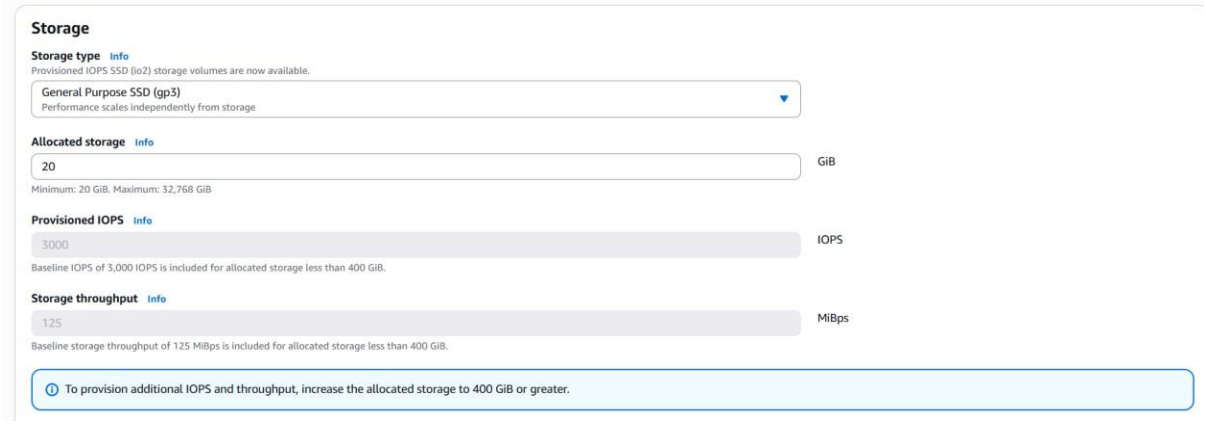
Instance Sizing Confirmed.

This image confirms the **DB instance class** selection within the configuration settings. The chosen class is **db.t3.medium**, categorized under **Burstable classes**. This selection adheres to the project's **minimum requirement** specification, providing **2 vCPUs** and **4 GB RAM** for the database instance.



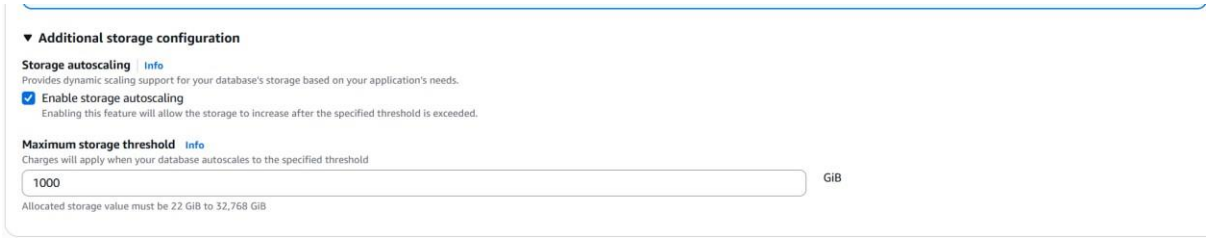
Storage Type Specified.

This screenshot confirms the **Storage configuration** for the RDS instance. The **Storage type** is set to **General Purpose SSD (gp3)**, offering high performance and independent scaling. The **Allocated storage** is set to **20 GiB**, and the default **Provisioned IOPS (3000)** and **Storage throughput (125 MiBps)** are utilized. This adheres to using modern, performant SSD storage.



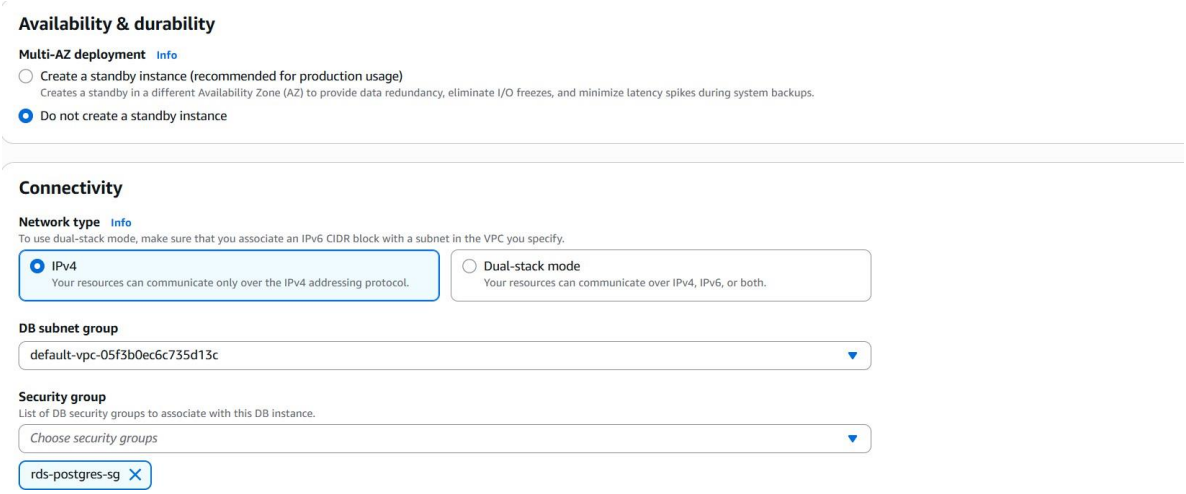
Storage Autoscaling Enabled.

This image shows the **Additional storage configuration** where **Storage autoscaling** is **Enabled**. This feature allows the database storage to dynamically increase when needed, up to the set **Maximum storage threshold** of **1000 GiB**. This ensures the database can handle growth without manual intervention.



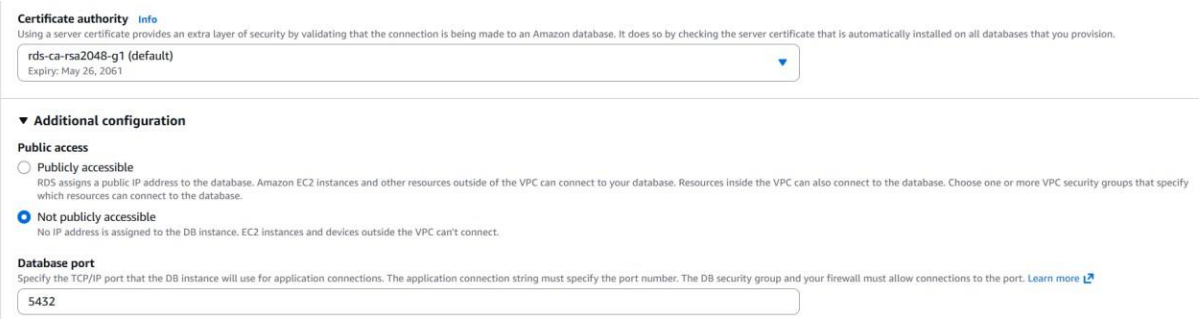
Connectivity Settings Chosen.

This screenshot displays the **Connectivity** configuration. It shows the **IPv4** network type selected and highlights the assignment of a **DB subnet group** (default VPC subnet group is used here). Critically, a **Security group** named rds-postgres-sg has been attached, which will control network access to the database instance. **Multi-AZ deployment** is currently set to **Do not create a standby instance**, but the project requires this to be enabled for production.



Security and Network Access.

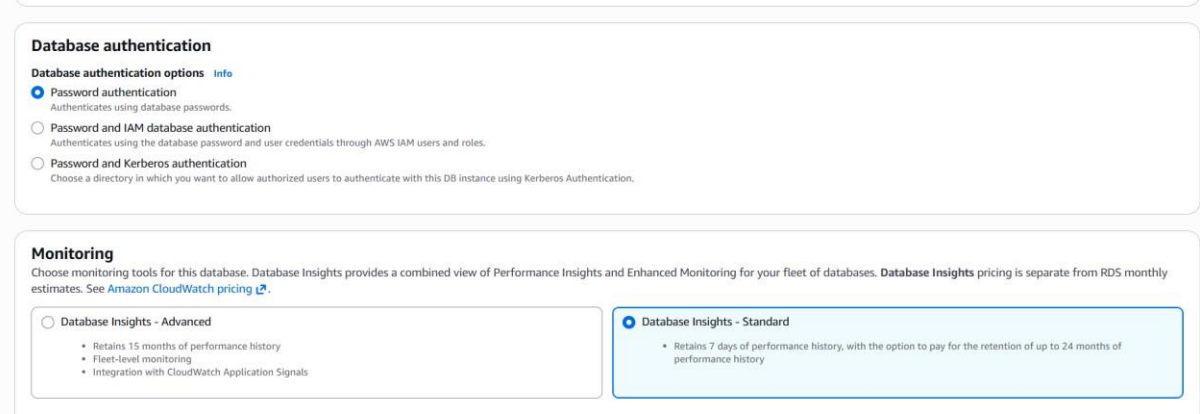
This image focuses on the **Additional configuration** for network security. It confirms that **Public access** is set to **Not publicly accessible**, ensuring the database is isolated within the VPC. The default **Database port** is correctly set to **5432 (PostgreSQL)**, and a **Certificate authority** is utilized for secure, validated connections (TLS/SSL).



Authentication and Monitoring Chosen.

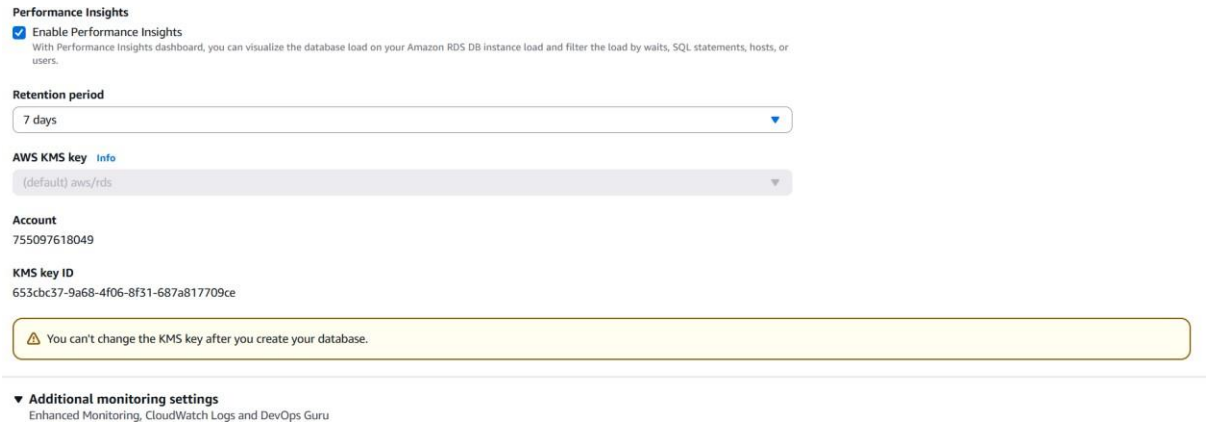
This image confirms two key configurations:

1. **Database Authentication: Password authentication** is selected, meaning users authenticate using traditional database passwords (managed manually or via Secrets Manager).
2. **Monitoring: Database Insights - Standard** is chosen. This option retains **7 days of performance history**, providing necessary metrics for basic monitoring and troubleshooting.



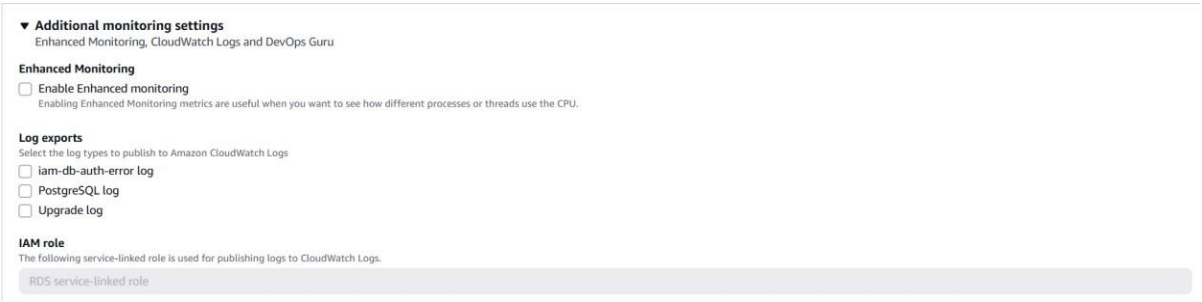
Performance Insights Enabled.

This screenshot confirms the configuration of **Performance Insights**, which is **Enabled** to visualize database load. The **Retention period** is set to **7 Days**. Importantly, it shows the database is configured to use an **AWS KMS key** ((default) aws/rds), ensuring that the database is **encrypted at rest** as required by security best practices



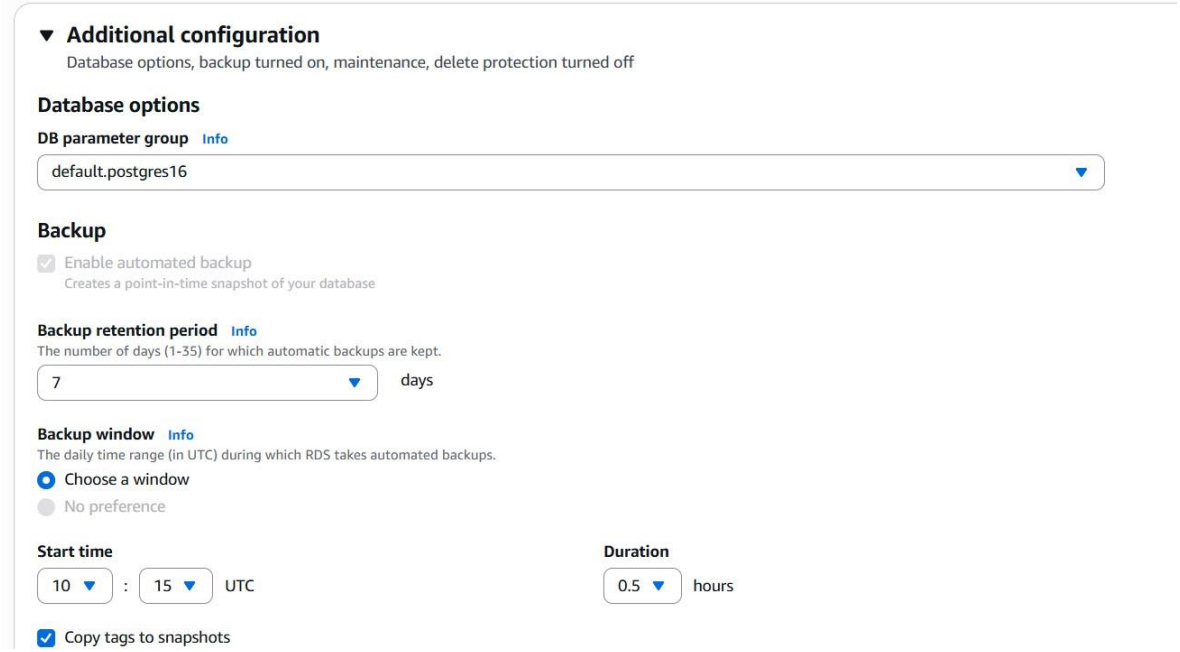
Additional Monitoring Settings.

This image displays the **Additional monitoring settings** for the RDS instance. It shows that **Enhanced Monitoring** is currently **disabled**. Below this, it provides options for **Log exports** (IAM, PostgreSQL, Upgrade) to Amazon CloudWatch Logs, none of which are currently selected. This section documents the default or chosen settings for advanced observability and logging



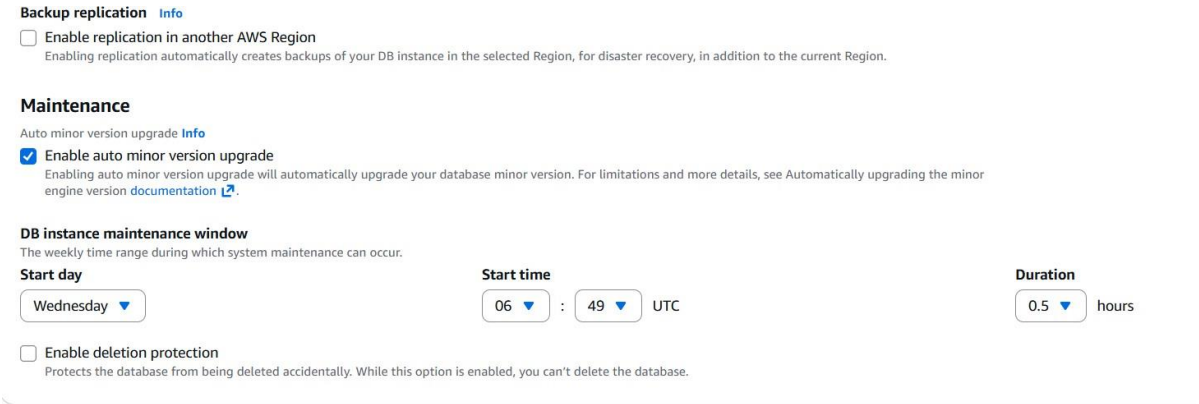
Backup and Retention Settings.

This image shows the **Backup configuration** under **Additional configuration**. It specifies that the **automated backup** is **not yet enabled** (checkbox is unchecked), though the **Backup retention period** is correctly set to **7 days** (meeting the project's minimum requirement). A custom **Backup window** is defined to start at **10:15 UTC** for a duration of **0.5 hours**. To comply with the project's requirements, automated backup **must be enabled** here.



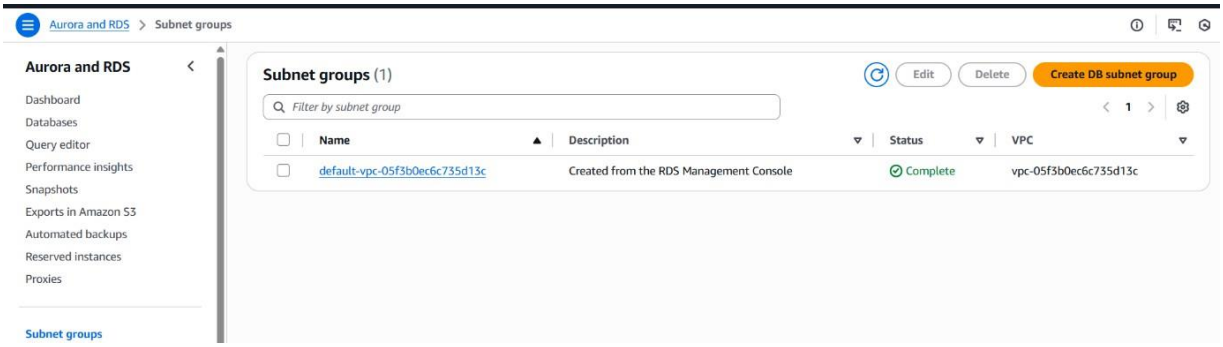
Maintenance and Protection Settings.

This final configuration image shows the **Maintenance** options. **Auto minor version upgrade** is **Enabled**, ensuring the database automatically receives small patches. The **DB instance maintenance window** is defined for **Wednesday at 06:4G UTC** for a **0.5-hour** duration. Finally, **Delete protection** is currently **disabled** (checkbox unchecked), meaning the database can be accidentally or intentionally deleted without an extra confirmation layer.



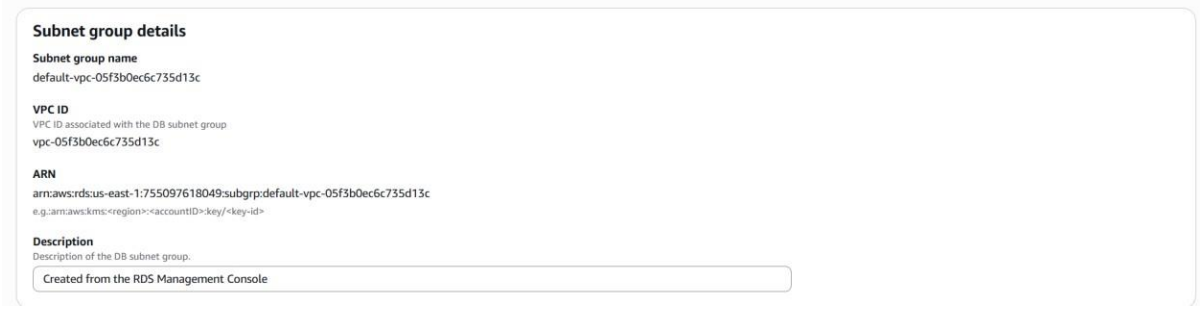
DB Subnet Group Verification.

This final screenshot confirms the existence of a **DB subnet group** named default-vpc-05f3b0ec6c735d13c with a status of **Complete**. A DB Subnet Group is essential for RDS to launch instances (especially Multi-AZ deployments) within your specified **VPC's private subnets** (Data Subnets, as per your project's architecture). This verifies the successful networking preparation needed for launching highly available databases.



Subnet Group Details.

This screenshot shows the detailed information for the **DB Subnet Group**. The **Subnet group name** is displayed, confirming its association with the specified **VPC ID (vpc-05f3b0ec6c735d13c)**. This group is essential for launching the RDS instance into the correct **private data subnets** within your VPC, a foundational requirement for securing the database and enabling **Multi-AZ deployments**.



Subnets Summary in DB Group.

This final screenshot provides a detailed table summarizing the 8 subnets selected for the DB Subnet Group. This is crucial documentation for the Configurations and Results sections, as it verifies the entire network architecture for the RDS deployment.

It shows that all four required subnet types (kafka, private, data, public) are correctly included across two Availability Zones (us-east-1a and us-east-1b), along with their associated CIDR blocks. This confirms that the RDS instance can be launched into the private data subnets (10.0.20.0/24 and 10.0.21.0/24) and supports the Multi-AZ requirement for high availability

Subnets selected (8)

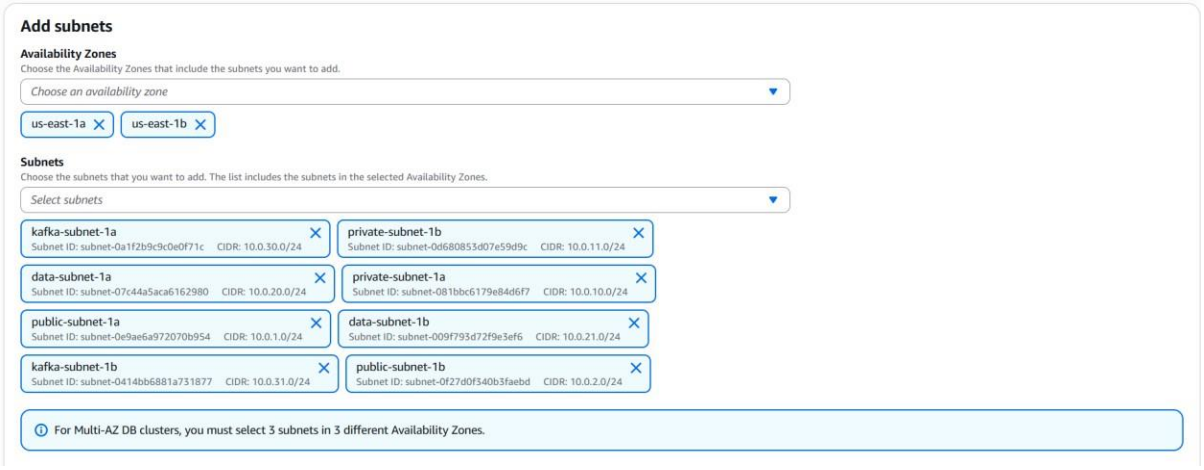
Availability zone	Subnet name	Subnet ID	CIDR block
us-east-1a	kafka-subnet-1a	subnet-0a1f2b9c9c0e0f71c	10.0.30.0/24
us-east-1b	private-subnet-1b	subnet-0d680853d07e59d9c	10.0.11.0/24
us-east-1a	data-subnet-1a	subnet-07c44a5aca6162980	10.0.20.0/24
us-east-1a	private-subnet-1a	subnet-081bbc6179e84d6f7	10.0.10.0/24
us-east-1a	public-subnet-1a	subnet-0e9ae6a972070b954	10.0.1.0/24
us-east-1b	data-subnet-1b	subnet-009f793d72f9e3ef6	10.0.21.0/24
us-east-1b	kafka-subnet-1b	subnet-0414bb6881a731877	10.0.31.0/24
us-east-1b	public-subnet-1b	subnet-0f27d0f340b3faebd	10.0.2.0/24

Subnets Selected for DB Group.

This screenshot shows the **"Add subnets"** configuration step for the DB Subnet Group. It confirms that subnets across **two Availability Zones (us-east-1a and us-east-1b)** have been selected. Specifically, the list includes all four required subnet types, categorized by AZ and CIDR blocks:

- **Kafka:** 10.0.30.0/24 and 10.0.31.0/24
- **Data:** 10.0.20.0/24 and 10.0.21.0/24
- **Private:** 10.0.10.0/24 and 10.0.11.0/24
- **Public:** 10.0.1.0/24 and 10.0.2.0/24

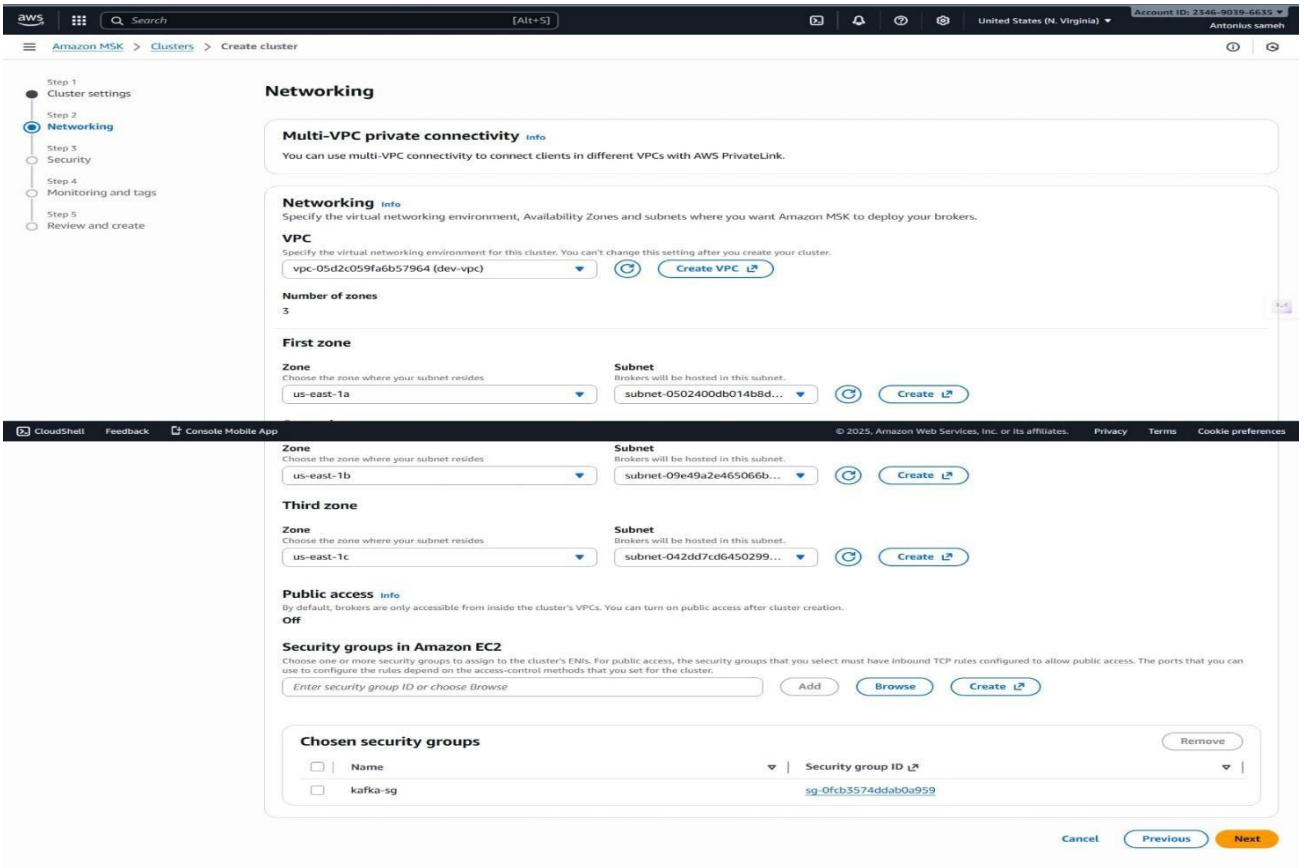
This step successfully collects the necessary subnets within the VPC to allow RDS to launch the database instance in the highly available **Data Subnets** across multiple AZs.



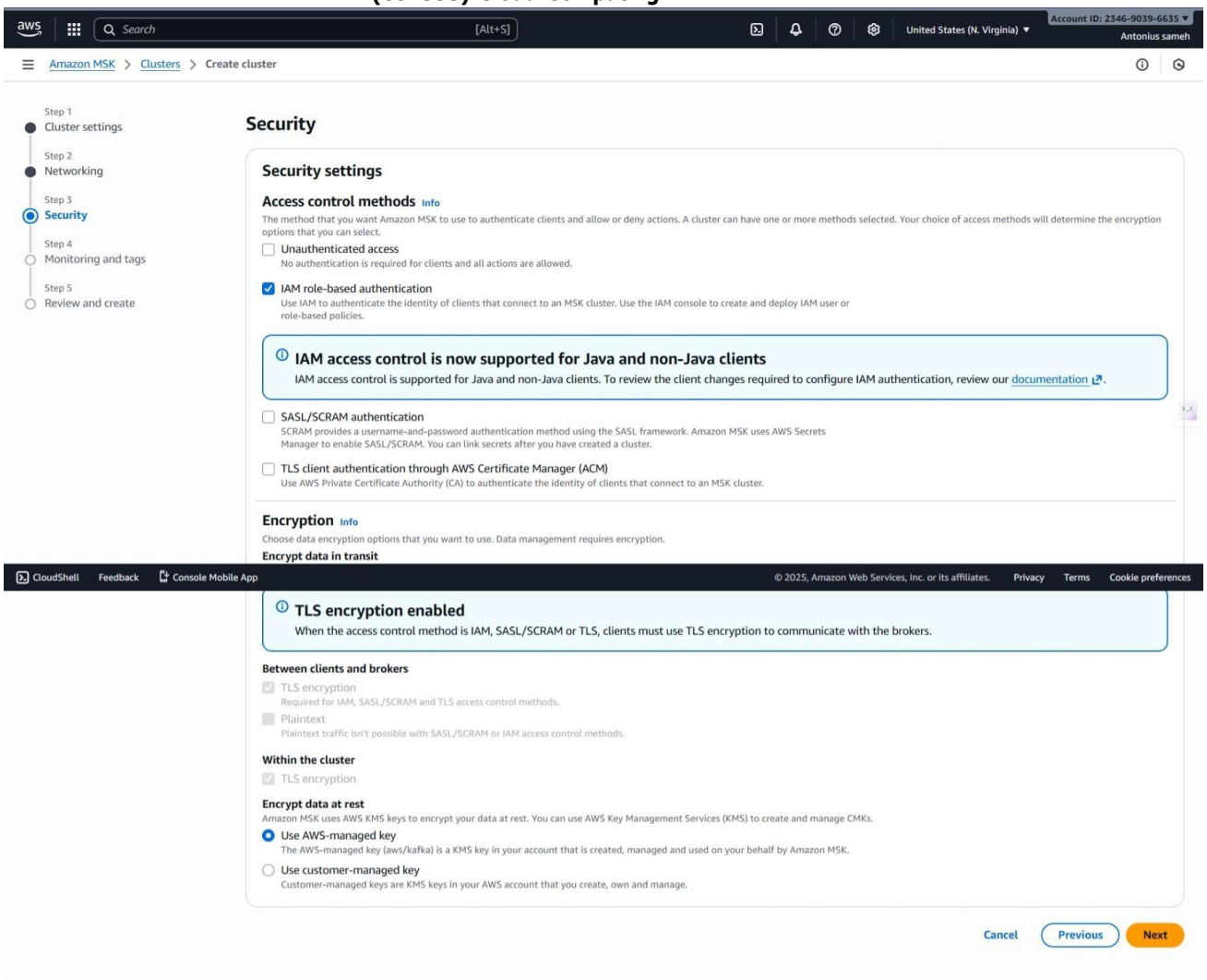
Phase 2

3. Apache Kafka

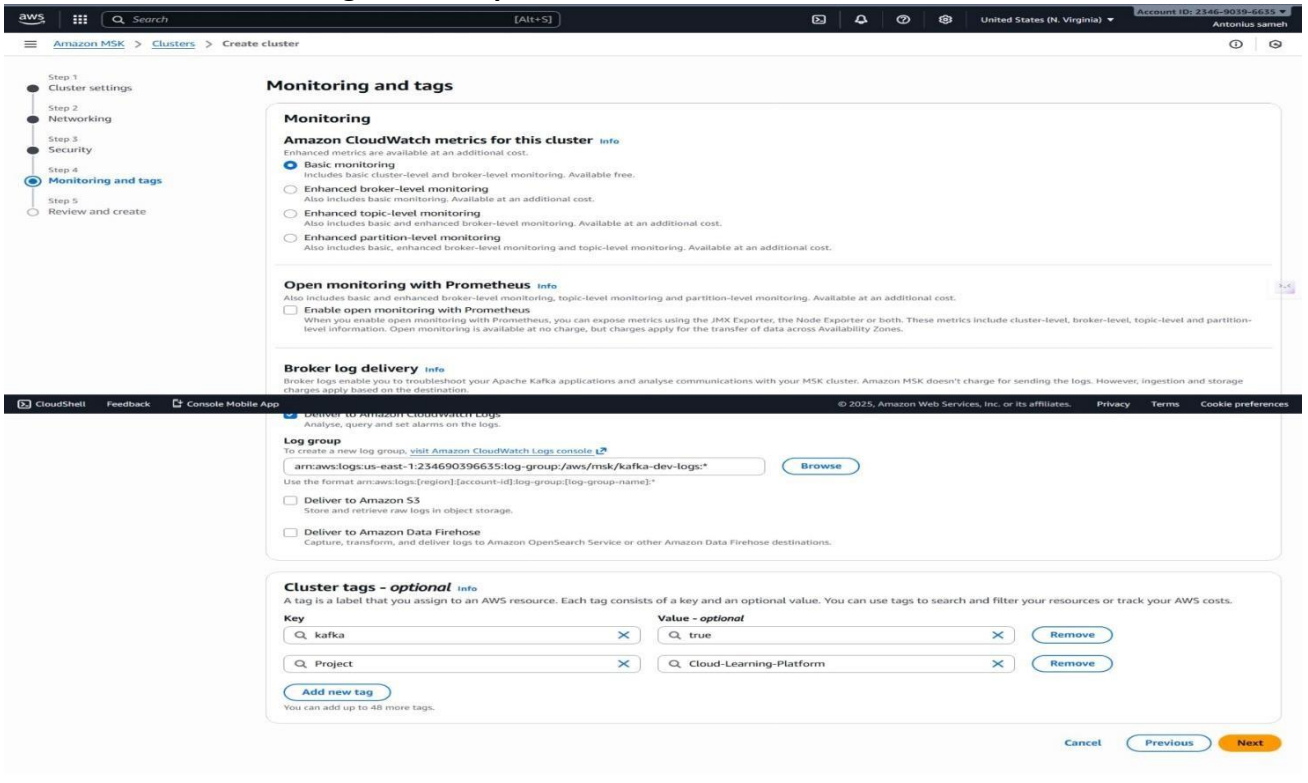
3.1 Kafka Cluster Architecture



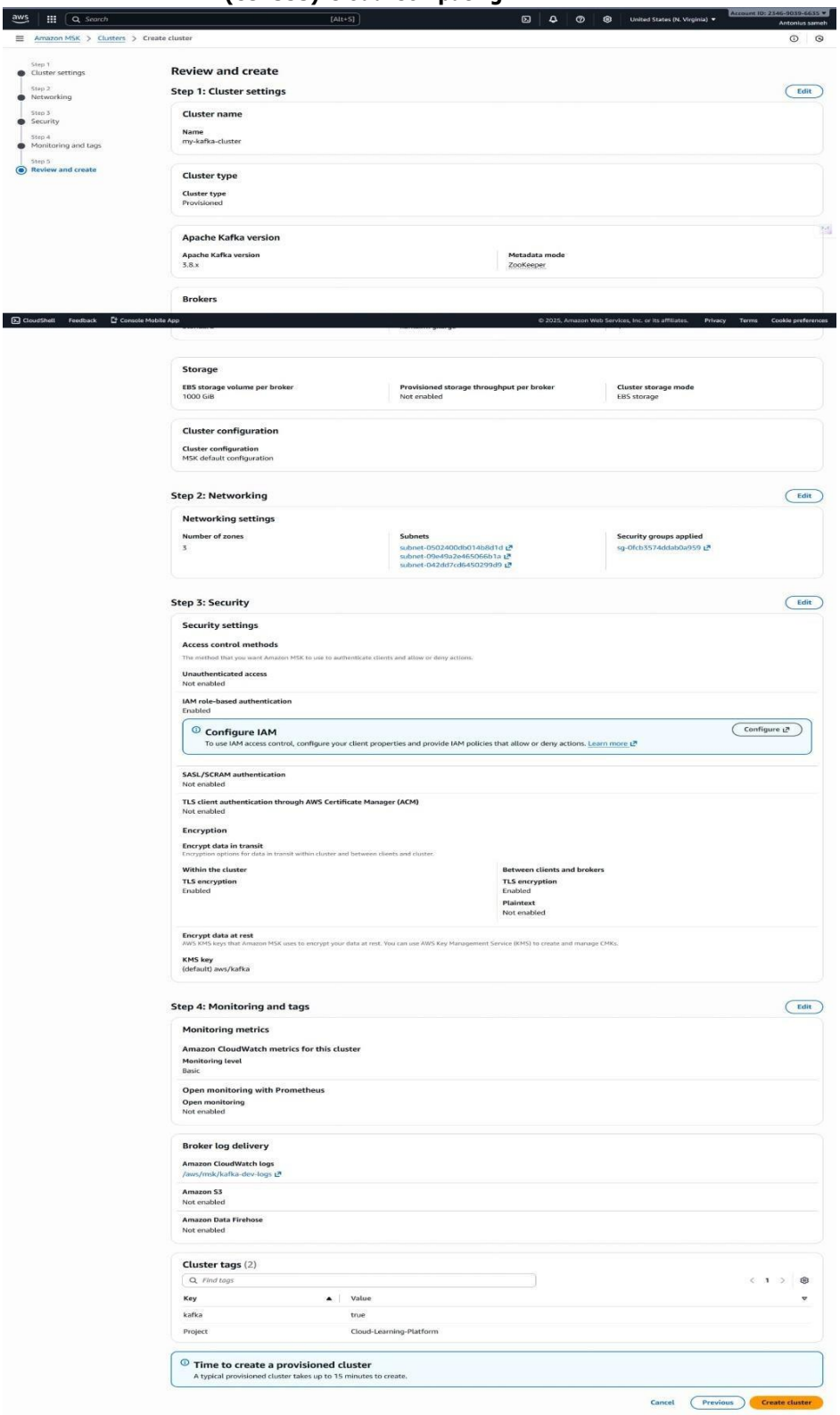
Networking settings for Amazon MSK cluster, including VPC, subnets across three zones, and security group assignment



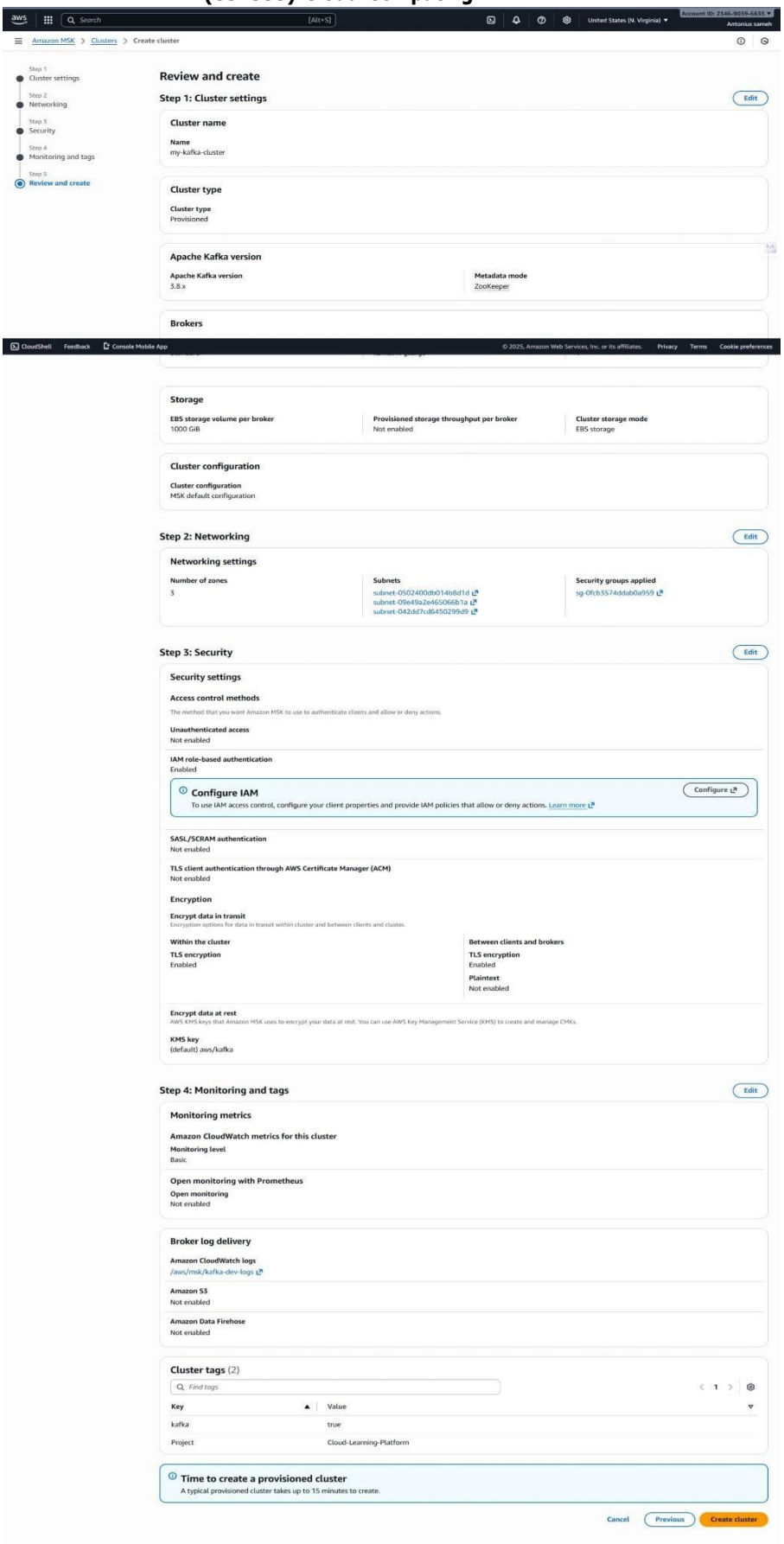
MSK cluster configured with private subnets in us-east-1a, 1b, and 1c zones.



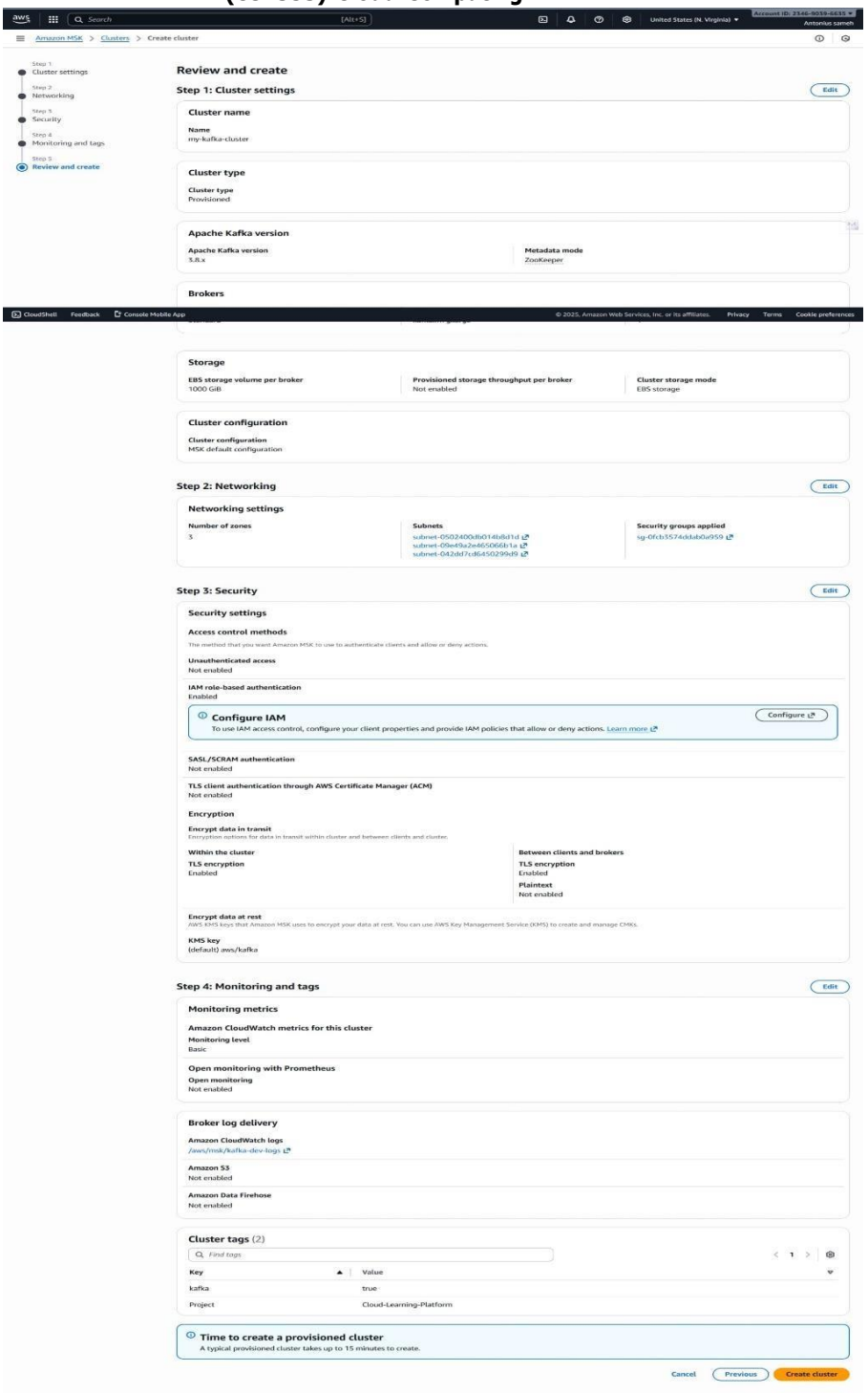
Basic CloudWatch monitoring enabled with tagged MSK cluster for project tracking.



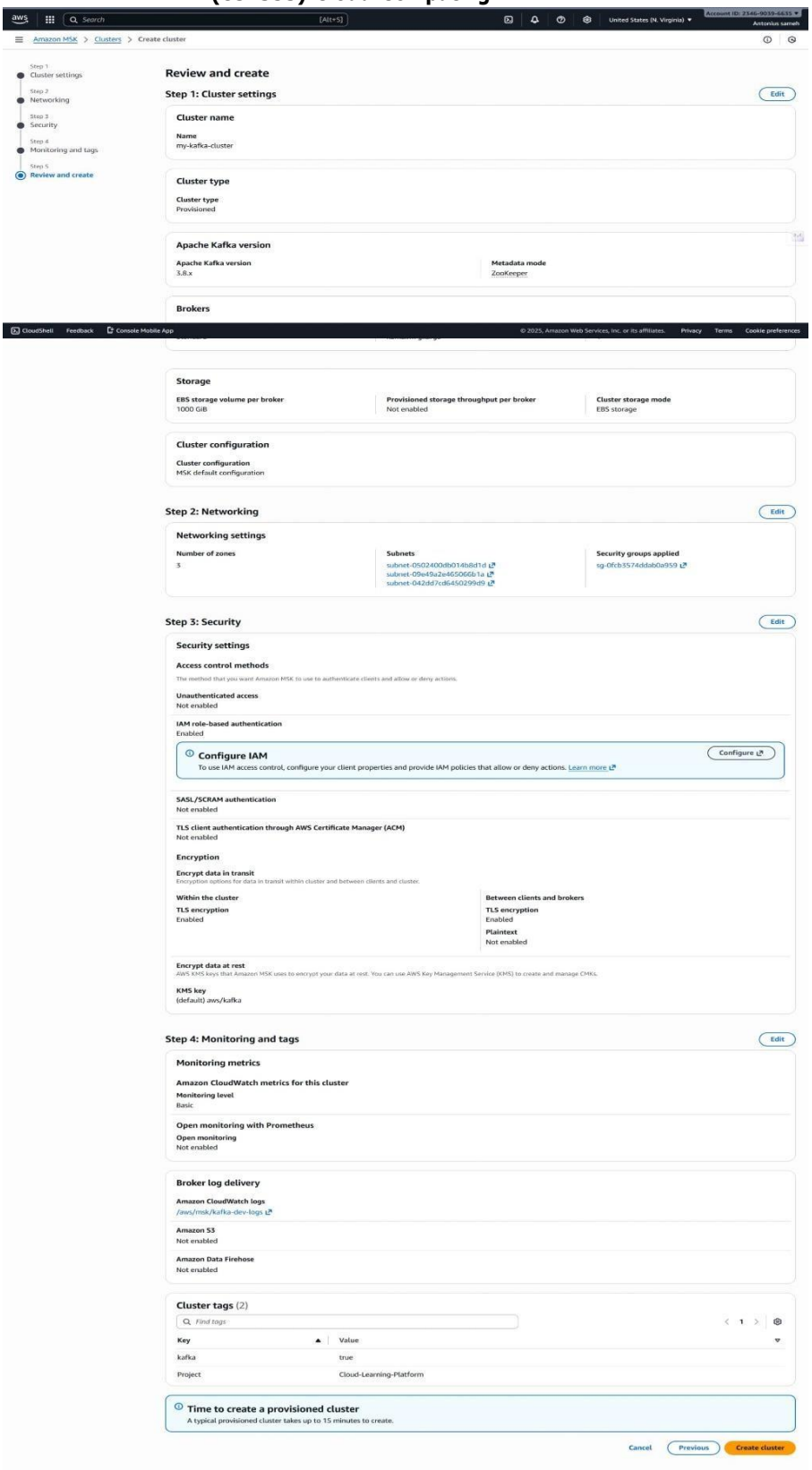
Final MSK cluster review: 3-zone setup, IAM security, TLS encryption, and basic monitoring.



Final MSK cluster review: 3-zone setup, IAM security, TLS encryption, and basic monitoring.



Final MSK cluster review: 3-zone setup, IAM security, TLS encryption, and basic monitoring.



Final MSK cluster review: 3-zone setup, IAM security, TLS encryption, and basic monitoring.

```
sh-5.2$ CLASSPATH=/home/ssm-user/aws-msk-iam-auth-1.1.9-all.jar \
/home/ssm-user/kafka_2.13-3.5.1/bin/kafka-topics.sh \
  --list \
  --bootstrap-server b-1.mykafkacluster.x5xpry.c21.kafka.us-east-1.amazonaws.com:9098 \
  --command-config /home/ssm-user/client.properties
__amazon_msk_canary
__consumer_offsets
audio.generation.completed
audio.generation.requested
audio.transcription.completed
audio.transcription.requested
chat.message
document.processed
document.uploaded
notes.generated
quiz.generated
quiz.requested
sh-5.2$
```

Kafka topics via MSK cluster using IAM and config file reference

Session ID: root-c5u5u5fhf35cfqzhodpric4y84 | Shortcuts | Instance ID: i-05dc18abf3d2df716 | Terminate

```
sh-5.2$ JAR="/home/ssm-user/aws-msk-iam-auth-1.1.9-all.jar"
KAFKA_BIN="/home/ssm-user/kafka_2.13-3.5.1/bin"
BOOT="b-1.mykafkacluster.x5xpry.c21.kafka.us-east-1.amazonaws.com:9098"
CFG="/home/ssm-user/client.properties"

echo "=== All Topics Details ==="
for topic in document.uploaded document.processed notes.generated quiz.requested quiz.generated audio.transcription.requested audio.transcription.completed audio.generation.
requested audio.generation.completed chat.message; do
  echo ""
  echo "--- Topic: $topic ---"
  CLASSPATH="$JAR" "$KAFKA_BIN/kafka-topics.sh" --describe --topic "$topic" --bootstrap-server "$BOOT" --command-config "$CFG"
done
=== All Topics Details ===

--- Topic: document.uploaded ---
Topic: document.uploaded        TopicId: K_fRoIX8Rb-oJHmQJpjLfQ PartitionCount: 3       ReplicationFactor: 2    Configs: min.insync.replicas=2,message.format.version=3.0-IV1
,unclean.leader.election.enable=true
        Topic: document.uploaded        Partition: 0    Leader: 3       Replicas: 3,2   Isr: 3,2
        Topic: document.uploaded        Partition: 1    Leader: 1       Replicas: 1,3   Isr: 1,3
        Topic: document.uploaded        Partition: 2    Leader: 2       Replicas: 2,1   Isr: 2,1

--- Topic: document.processed ---
Topic: document.processed       TopicId: gJa5TcjrSsy9V_HIoT5jew PartitionCount: 3       ReplicationFactor: 2    Configs: min.insync.replicas=2,message.format.version=3.0-IV1
,unclean.leader.election.enable=true
        Topic: document.processed       Partition: 0    Leader: 1       Replicas: 1,2   Isr: 1,2
        Topic: document.processed       Partition: 1    Leader: 2       Replicas: 2,3   Isr: 2,3
        Topic: document.processed       Partition: 2    Leader: 3       Replicas: 3,1   Isr: 3,1

--- Topic: notes.generated ---
Topic: notes.generated  TopicId: pxTJOvdkQ9CWMfI2lniOJg PartitionCount: 2       ReplicationFactor: 2    Configs: min.insync.replicas=2,message.format.version=3.0-IV1,unclean
.leader.election.enable=true
        Topic: notes.generated  Partition: 0    Leader: 1       Replicas: 1,2   Isr: 1,2
        Topic: notes.generated  Partition: 1    Leader: 2       Replicas: 2,3   Isr: 2,3

--- Topic: quiz.requested ---
Topic: quiz.requested   TopicId: 2couly6nS8ORytl4ohlZvQ PartitionCount: 2       ReplicationFactor: 2    Configs: min.insync.replicas=2,message.format.version=3.0-IV1,unclean
```

Described Kafka topics with partition count, replication factor, and in-sync replica configuration

Session ID: root-c5u5u5fhf35cfqzhodpric4y84 | Shortcuts | Instance ID: i-05dc18abf3d2df716 | Terminate

```
        Topic: quiz.generated   Partition: 0    Leader: 3       Replicas: 3,1   Isr: 3,1
        Topic: quiz.generated   Partition: 1    Leader: 1       Replicas: 1,2   Isr: 1,2

--- Topic: audio.transcription.requested ---
Topic: audio.transcription.requested    TopicId: VkyAIIe2RA2rTIk4-DQffA PartitionCount: 2       ReplicationFactor: 2    Configs: min.insync.replicas=2,message.format.version
=3.0-IV1,unclean.leader.election.enable=true
        Topic: audio.transcription.requested    Partition: 0    Leader: 1       Replicas: 1,3   Isr: 1,3
        Topic: audio.transcription.requested    Partition: 1    Leader: 2       Replicas: 2,1   Isr: 2,1

--- Topic: audio.transcription.completed ---
Topic: audio.transcription.completed    TopicId: D7WGO-GASqy3DvygscTUhw PartitionCount: 2       ReplicationFactor: 2    Configs: min.insync.replicas=2,message.format.version
=3.0-IV1,unclean.leader.election.enable=true
        Topic: audio.transcription.completed    Partition: 0    Leader: 1       Replicas: 1,3   Isr: 1,3
        Topic: audio.transcription.completed    Partition: 1    Leader: 2       Replicas: 2,1   Isr: 2,1

--- Topic: audio.generation.requested ---
Topic: audio.generation.requested       TopicId: t3xL-Do9RsqK7oM2AlOMWA PartitionCount: 2       ReplicationFactor: 2    Configs: min.insync.replicas=2,message.format.version
=3.0-IV1,unclean.leader.election.enable=true
        Topic: audio.generation.requested       Partition: 0    Leader: 3       Replicas: 3,1   Isr: 3,1
        Topic: audio.generation.requested       Partition: 1    Leader: 1       Replicas: 1,2   Isr: 1,2

--- Topic: audio.generation.completed ---
Topic: audio.generation.completed       TopicId: hYDRhhNASY6GWTvUV8bUYg PartitionCount: 2       ReplicationFactor: 2    Configs: min.insync.replicas=2,message.format.version
=3.0-IV1,unclean.leader.election.enable=true
        Topic: audio.generation.completed       Partition: 0    Leader: 2       Replicas: 2,1   Isr: 2,1
        Topic: audio.generation.completed       Partition: 1    Leader: 3       Replicas: 3,2   Isr: 3,2

--- Topic: chat.message ---
Topic: chat.message     TopicId: WctIJ9V8TI6vHhUGlIB01g PartitionCount: 6       ReplicationFactor: 2    Configs: min.insync.replicas=2,message.format.version=3.0-IV1,unclean
.leader.election.enable=true
        Topic: chat.message     Partition: 0    Leader: 3       Replicas: 3,1   Isr: 3,1
        Topic: chat.message     Partition: 1    Leader: 1       Replicas: 1,2   Isr: 1,2
        Topic: chat.message     Partition: 2    Leader: 2       Replicas: 2,3   Isr: 2,3
        Topic: chat.message     Partition: 3    Leader: 3       Replicas: 3,2   Isr: 3,2
        Topic: chat.message     Partition: 4    Leader: 1       Replicas: 1,3   Isr: 1,3
        Topic: chat.message     Partition: 5    Leader: 2       Replicas: 2,1   Isr: 2,1
```

Kafka CLI output confirms topic configurations with 3 partitions, 2 replicas, and active ISR sets

```
sh-5.2$ CLASSPATH="$JAR" "$KAFKA_BIN/kafka-console-consumer.sh" \
  --bootstrap-server "$BOOT" \
  --topic quiz.requested \
  --consumer.config "$CFG" \
  --group my-test-group \
  --from-beginning
hello test 1764863412
test message
hello-from-producer-1764862415
hello test 1764862775
hello test 1764863431
hello from producer
hi from producer
```

Messages published to 'quiz.requested' topic using MSK console producer

```
sh-5.2$ CLASSPATH="$JAR" "$KAFKA_BIN/kafka-console-producer.sh" \
  --broker-list "$BOOT" \
  --topic quiz.requested \
  --producer.config "$CFG"
>hello from producer
>hi from producer
>
```

MSK console consumer connected to topic 'quiz.requested' and received messages successfully.

3.2 Kafka Integration Patterns

1. Event Sourcing

Purpose:

Track every state change as an immutable sequence of events instead of storing only the final state.

How it is implemented:

- Services produce events to Kafka (e.g., document.events, quiz.requested).
- Consumers rebuild state from these events.

Benefits:

- Full audit/history
- Debugging and replay
- Supports rebuilding materialized views

2. CQRS

Purpose:

Separate write operations (commands) from read operations (queries).

How it is implemented:

- Write side publishes events (e.g., audio.transcription.requested).
- Read side consumes and builds query models.

Benefits:

- Scalable reads
- Independent scaling
- Cleaner boundaries

3. Saga Pattern

Purpose:

Handle multi-step transactions across microservices.

How it is implemented:

- Orchestrator uses saga.requests.
- Services respond with success/failure events.
- Compensating events undo failures.

Benefits:

- Reliable workflows
- Rollback supported
- Loose coupling

4. Event Notification

Purpose:

Notify services of significant events.

How it is implemented:

- Services publish notifications (e.g., notifications.document).

Benefits:

- Loose coupling
- Real-time updates
- Easy extensibility

---

4. API Gateway Implementation

4.1 API Gateway Architecture

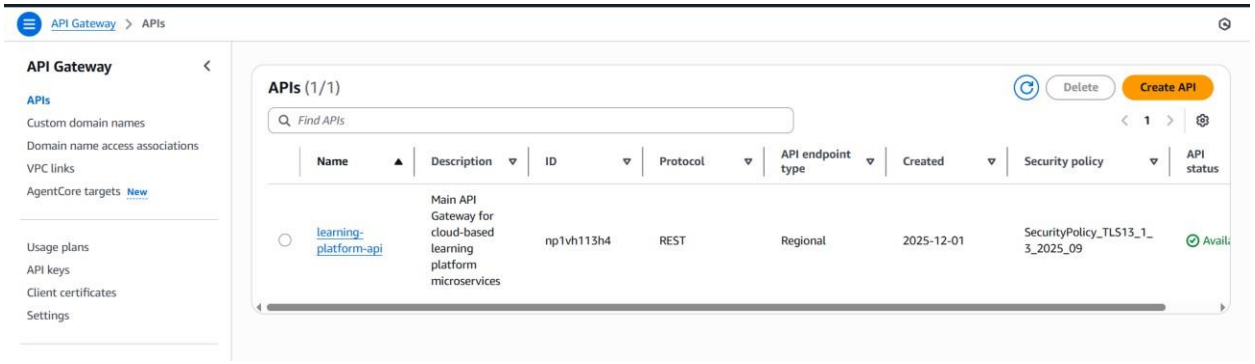
This guide outlines the chronological sequence of steps executed to establish a secure, scalable API Gateway using AWS services, fulfilling all technical and documentation requirements for the project.

I. Gateway Foundation and Routing Setup

This phase established the single entry point and defined the mechanism for routing incoming traffic to the backend microservices (running behind the ALB).

Step 1: API Gateway Creation and Stage Setup

1. **Implementation commenced** with the deployment of the **AWS API Gateway (REST API)**, named `learning-platform-api`.
2. **Purpose:** This API was established as the central point of control for the entire application traffic.

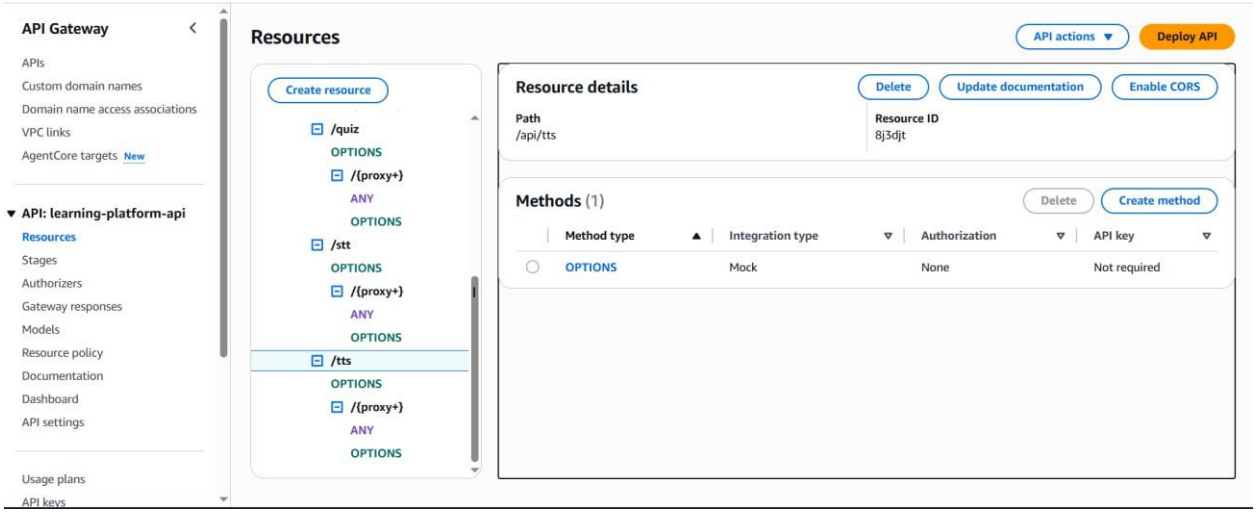


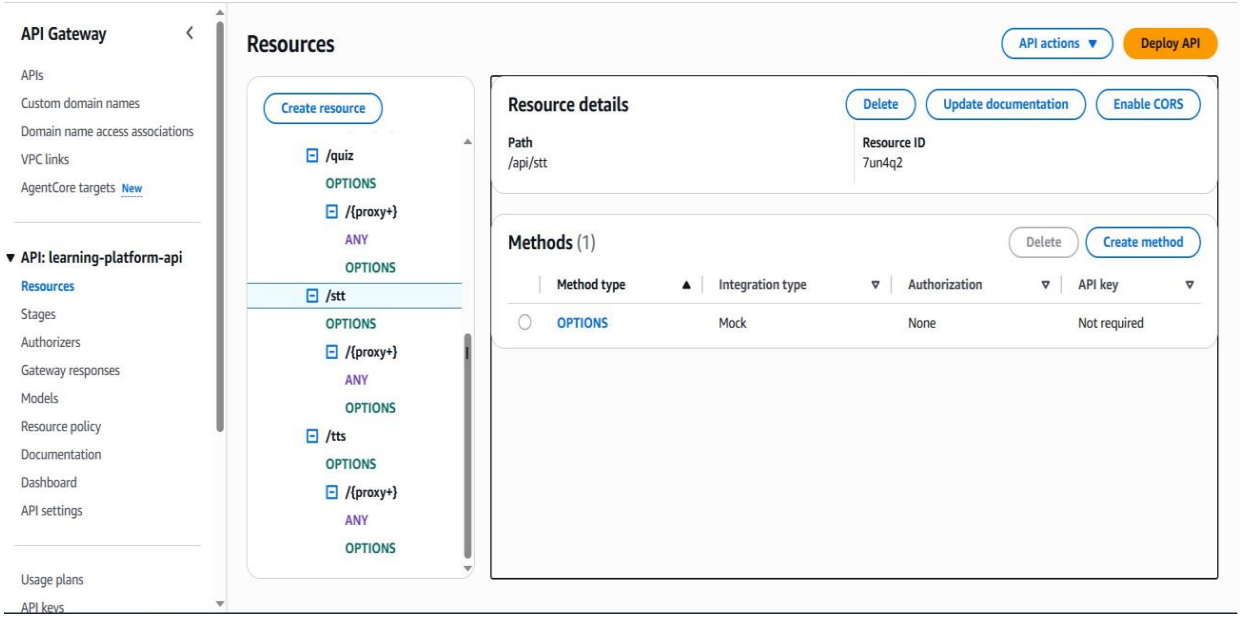
Step 2: Resource and Proxy Configuration

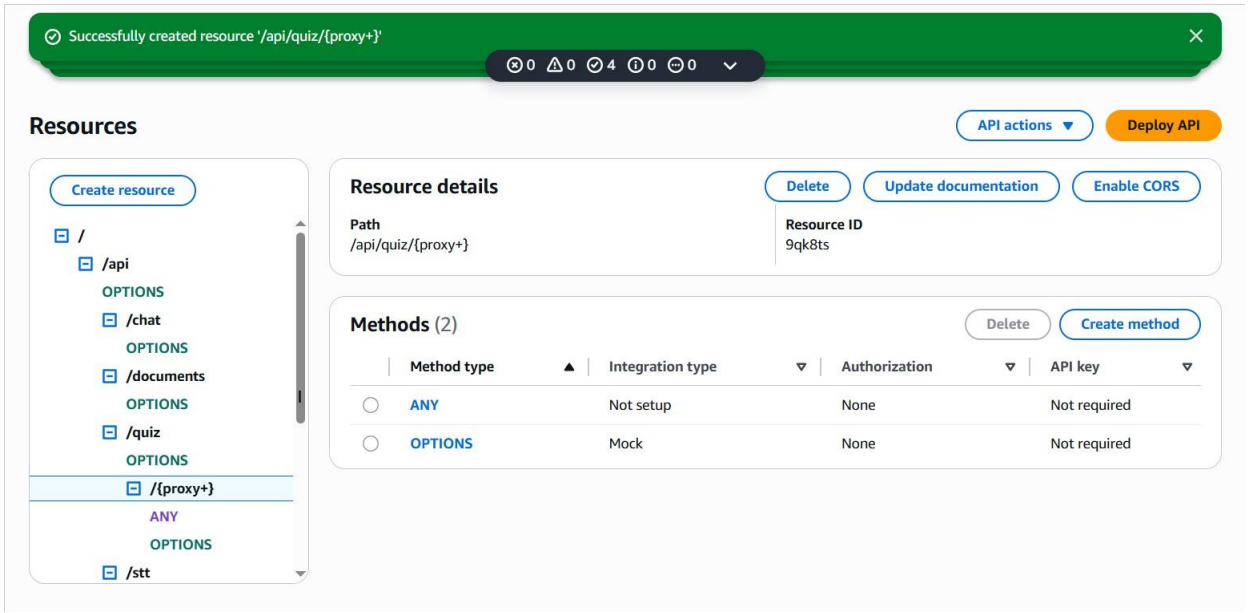
1. All primary service path structures (e.g., `/api/tts`, /api/documents, /api/chat, etc.) were established as resources under the main /apiroot.

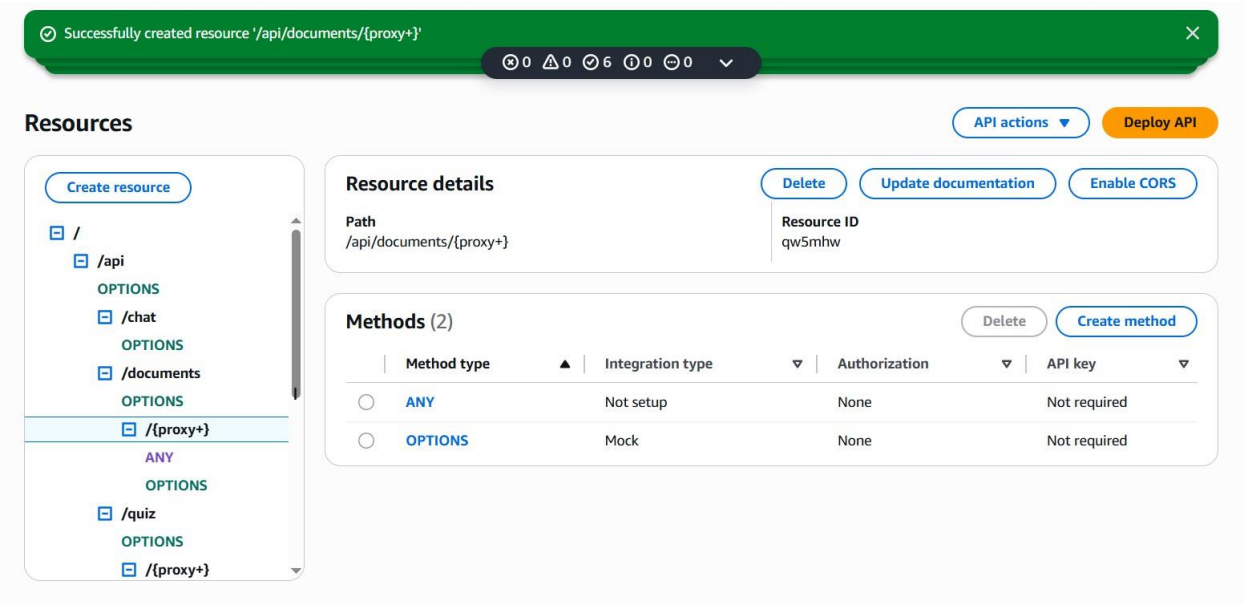
2. To enable comprehensive routing, a child resource **`/{proxy+}`** was created for each service path, which is mandatory for **Proxy Integration** to ensure all underlying requests (e.g., /api/tts/synthesize) are handled correctly.

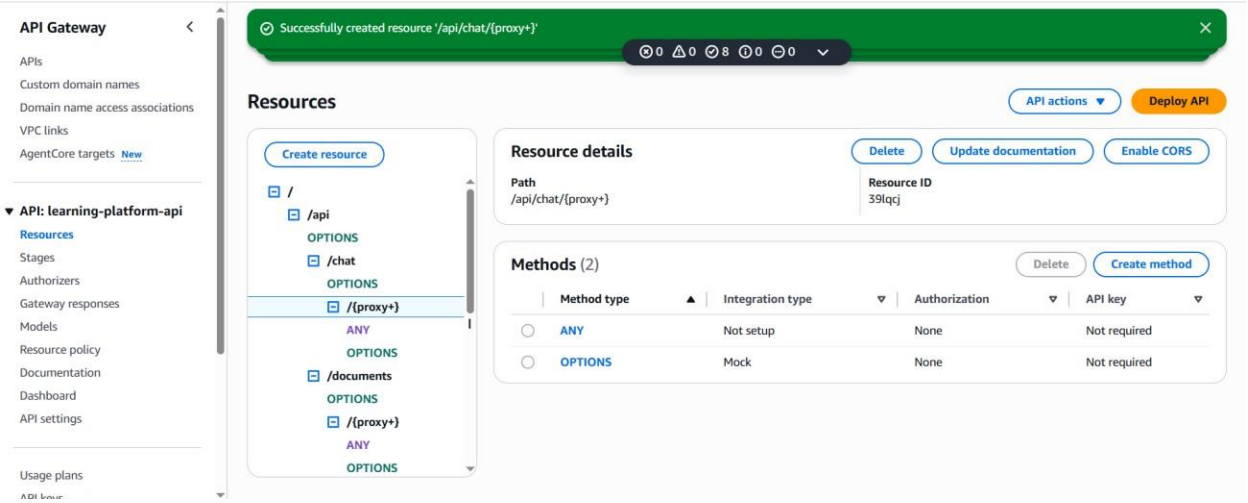
3. **CORS (Cross-Origin Resource Sharing)** was globally enabled on all resources to facilitate frontend connectivity.


API Gateway
APIs
Custom domain names
Domain name access associations
VPC links
AgentCore targets New
API: learning-platform-api
Resources
Stages
Authorizers
Gateway responses
Models
Resource policy
Documentation
Dashboard
API settings
Usage plans
API keys
Resources
API actions
Deploy API
Create resource
/quiz
OPTIONS
/{proxy+}
ANY
OPTIONS
/stt
OPTIONS
/{proxy+}
ANY
OPTIONS
/tts
OPTIONS
/{proxy+}
ANY
OPTIONS
Resource details
Delete
Update documentation
Enable CORS
Path
/api/tts
Resource ID
8j3djt
Methods (1)
Delete
Create method
Method type
Integration type
Authorization
API key
OPTIONS
Mock
None
Not required


API Gateway
APIs
Custom domain names
Domain name access associations
VPC links
AgentCore targets New
API: learning-platform-api
Resources
Stages
Authorizers
Gateway responses
Models
Resource policy
Documentation
Dashboard
API settings
Usage plans
API keys
Resources
API actions
Deploy API
Create resource
/quiz
OPTIONS
/{proxy+}
ANY
OPTIONS
/stt
OPTIONS
/{proxy+}
ANY
OPTIONS
/tts
OPTIONS
/{proxy+}
ANY
OPTIONS
Resource details
Delete
Update documentation
Enable CORS
Path
/api/stt
Resource ID
7un4q2
Methods (1)
Delete
Create method
Method type
Integration type
Authorization
API key
OPTIONS
Mock
None
Not required

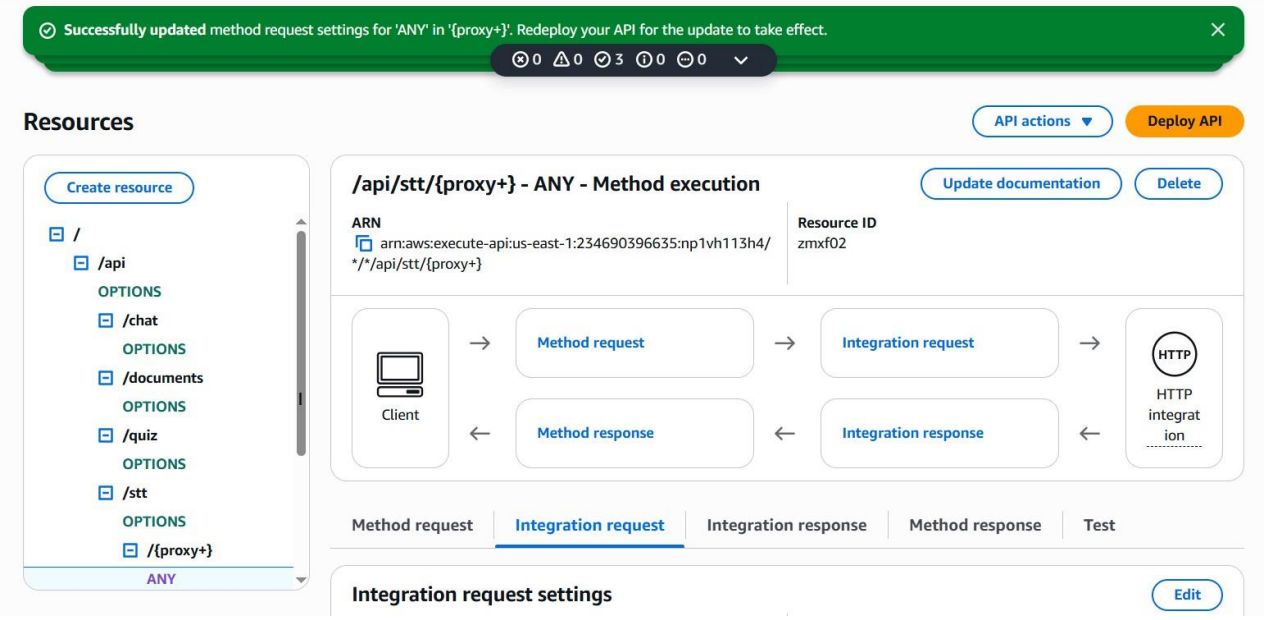

Successfully created resource '/api/quiz/{proxy+}'
Resources
API actions
Deploy API
Create resource
/
/api
OPTIONS
/chat
OPTIONS
/documents
OPTIONS
/quiz
OPTIONS
/{proxy+}
ANY
OPTIONS
/stt
Resource details
Delete
Update documentation
Enable CORS
Path
/api/quiz/{proxy+}
Resource ID
9qk8ts
Methods (2)
Delete
Create method
Method type
Integration type
Authorization
API key
ANY
Not setup
None
Not required
OPTIONS
Mock
None
Not required

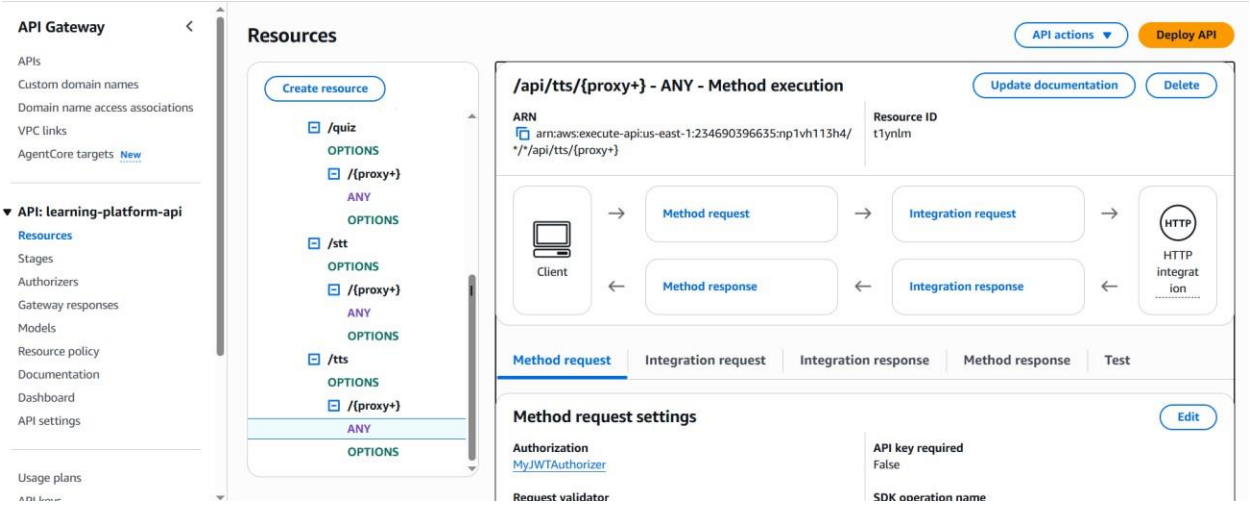


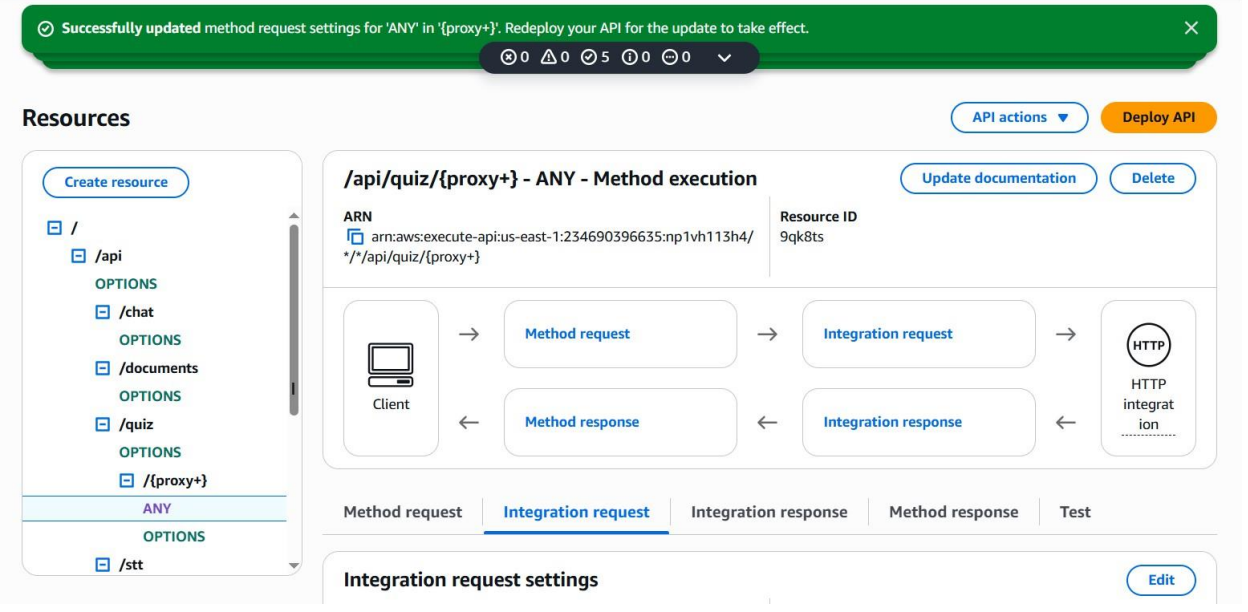


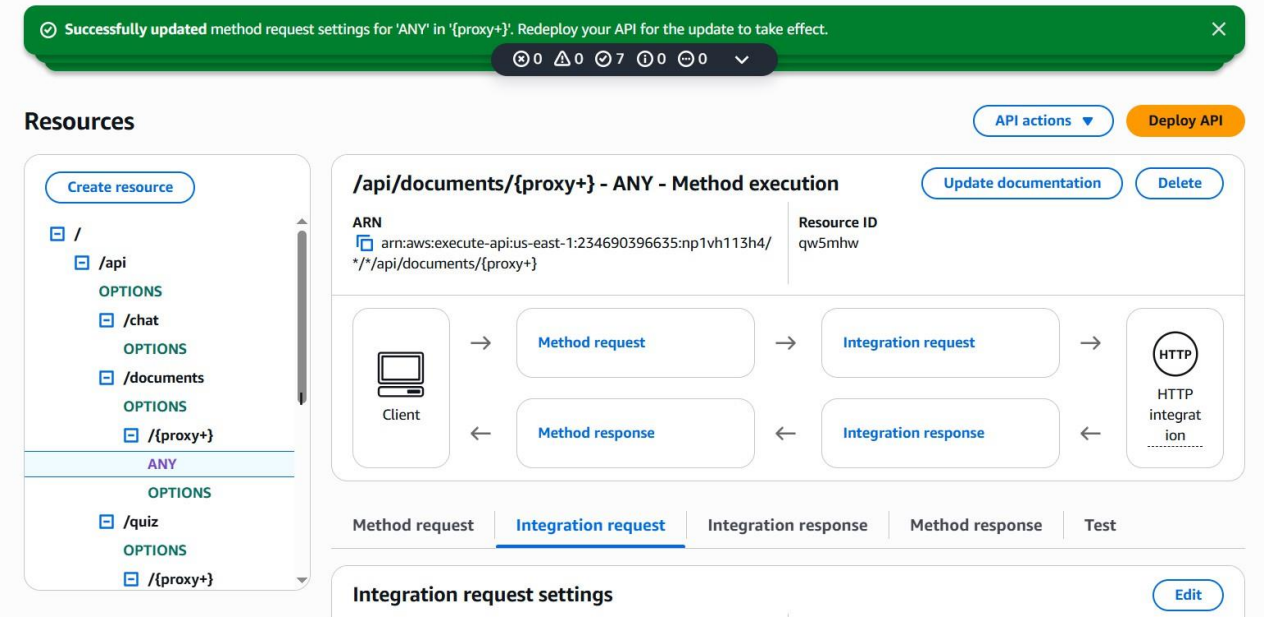
Step 3: Setting HTTP Proxy Integration

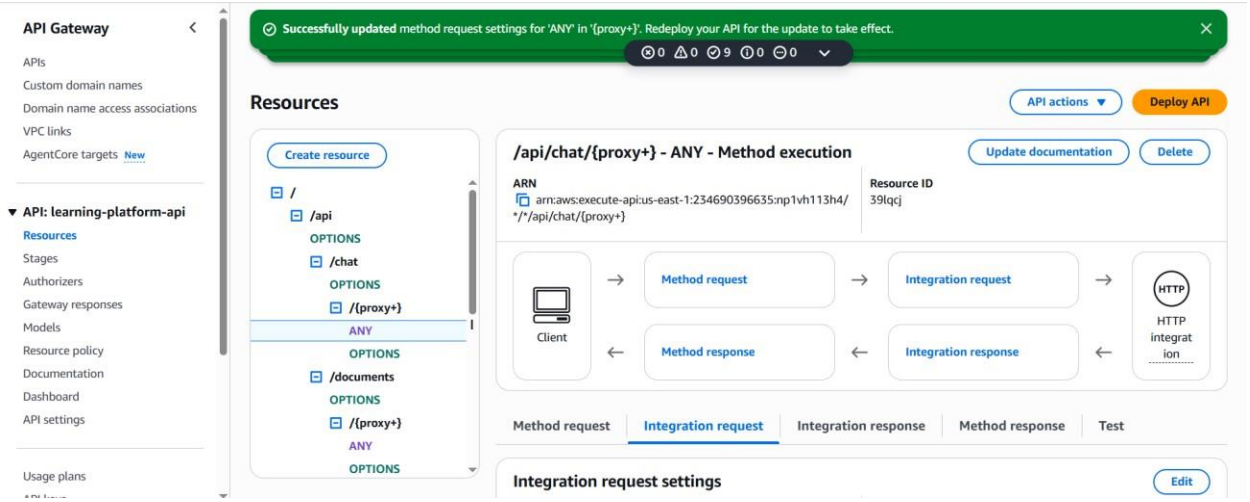
1. The **Integration Request** for the ANY method under each /{proxy+}resource was configured as **HTTP Proxy**.
2. The **Endpoint URL** was explicitly set to the **Application Load Balancer (ALB)** DNS name, including the required service path (e.g., http://dev-public-alb.../api/tts/{proxy}).
3. The **Content Handling** setting was maintained as **Passthrough** to ensure the gateway transmits raw data (file uploads, complex JSON) directly to the backend services without alteration.
4. **Output:** This step successfully linked the public API endpoint to the private backend ALB.


Successfully updated method request settings for 'ANY' in '{proxy+}'. Redeploy your API for the update to take effect.
Resources
API actions
Deploy API
Create resource
/
/api
OPTIONS
/chat
OPTIONS
/documents
OPTIONS
/quiz
OPTIONS
/stt
OPTIONS
/{proxy+}
ANY
/api/stt/{proxy+} - ANY - Method execution
Update documentation
Delete
ARN
arn:aws:execute-api:us-east-1:234690396635:np1vh113h4/*/*/api/stt/{proxy+}
Resource ID
zmxf02
Client
Method request
Integration request
HTTP
HTTP integration
Method response
Integration response
Method request
Integration request
Integration response
Method response
Test
Integration request settings
Edit


API Gateway
APIs
Custom domain names
Domain name access associations
VPC links
AgentCore targets New
API: learning-platform-api
Resources
Stages
Authorizers
Gateway responses
Models
Resource policy
Documentation
Dashboard
API settings
Usage plans
Resources
API actions
Deploy API
Create resource
/quiz
OPTIONS
/{proxy+}
ANY
OPTIONS
/stt
OPTIONS
/{proxy+}
ANY
OPTIONS
/tts
OPTIONS
/{proxy+}
ANY
OPTIONS
/api/tts/{proxy+} - ANY - Method execution
Update documentation
Delete
ARN
arn:aws:execute-api:us-east-1:234690396635:np1vh113h4/*/*/api/tts/{proxy+}
Resource ID
t1ynlm
Client
Method request
Integration request
HTTP
HTTP integration
Method response
Integration response
Method request
Integration request
Integration response
Method response
Test
Method request settings
Edit
Authorization
MyJWTAuthorizer
API key required
False
Request validator
SDK operation name


Successfully updated method request settings for 'ANY' in '{proxy+}'. Redeploy your API for the update to take effect.
Resources
API actions
Deploy API
Create resource
/
/api
OPTIONS
/chat
OPTIONS
/documents
OPTIONS
/quiz
OPTIONS
/{proxy+}
ANY
OPTIONS
/stt
/api/quiz/{proxy+} - ANY - Method execution
Update documentation
Delete
ARN
arn:aws:execute-api:us-east-1:234690396635:np1vh113h4/*/*/api/quiz/{proxy+}
Resource ID
9qk8ts
Client
Method request
Integration request
HTTP
HTTP integration
Method response
Integration response
Method request
Integration request
Integration response
Method response
Test
Integration request settings
Edit

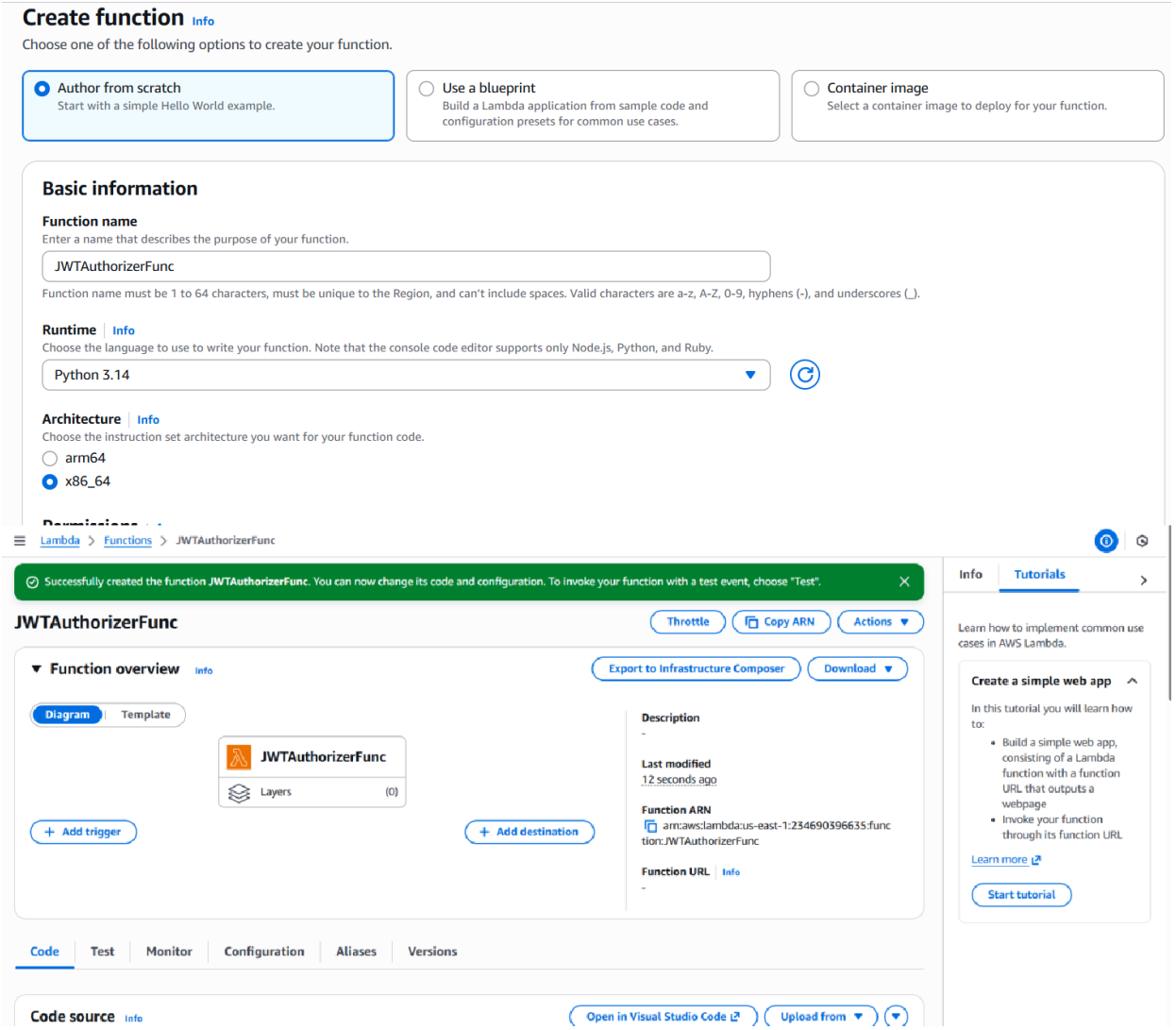




II. Security and Control Implementation

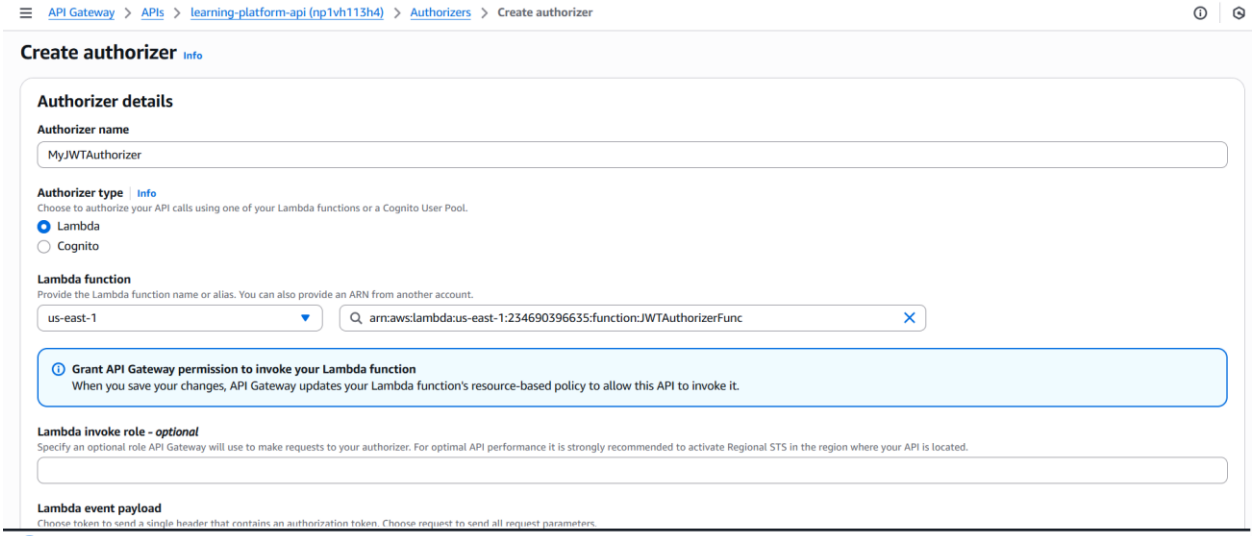
Step 4: Creating the Lambda Authorizer Function

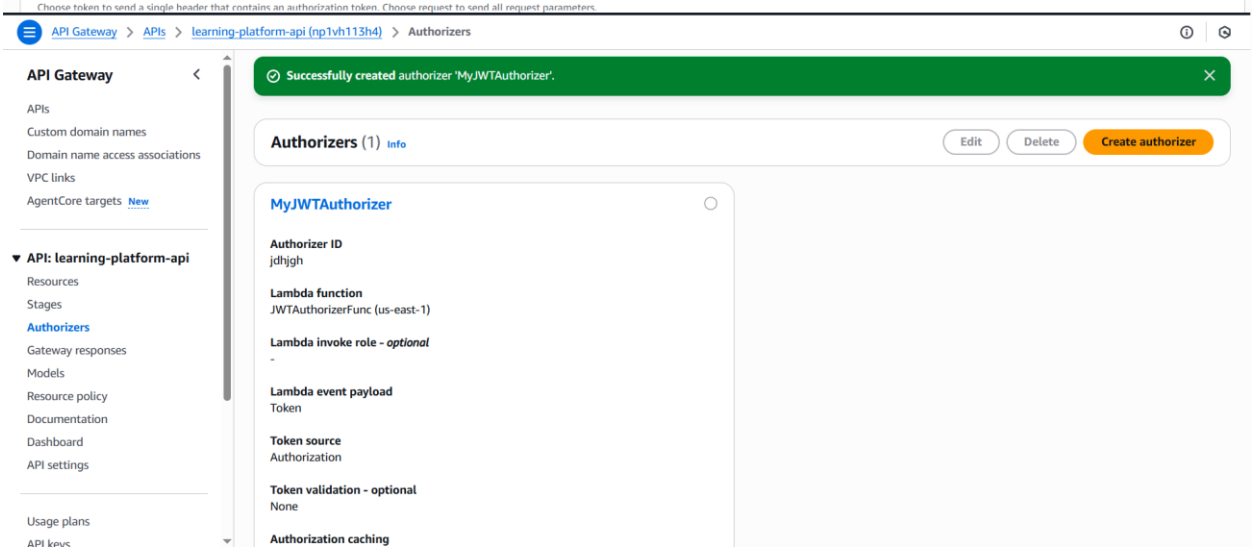
- A dedicated Lambda function named `JWTAuthorizerFunc` was created in the AWS Lambda console. This function was deployed with the necessary code to contain the **JWT validation logic**.
- **Purpose:** This function serves as the external security policy engine for the gateway.



Step 5: Defining the Authorizer in API Gateway

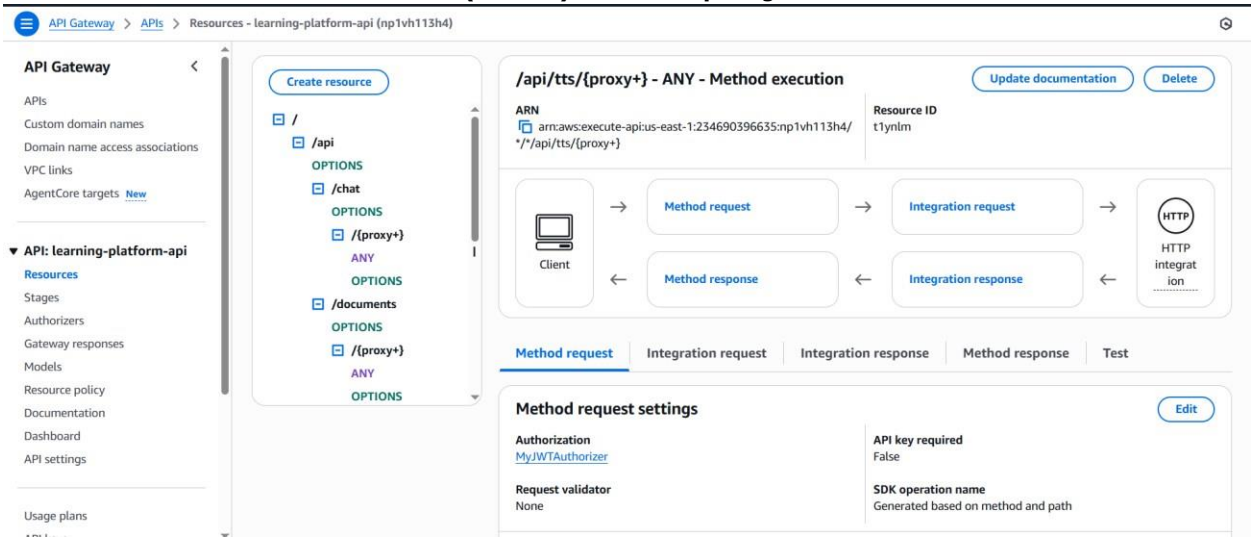
- A new **Lambda Authorizer** instance, named `MyJWTAuthorizer`, was defined in the API Gateway console and successfully linked to the `JWTAuthorizerFunc` from Step 4.
- The **Token source** was explicitly configured to read the **`Authorization`** request header, which is the standard location for receiving the JWT from the client.
- **Output:** The Gateway is now configured to invoke the Lambda function for authentication checks.





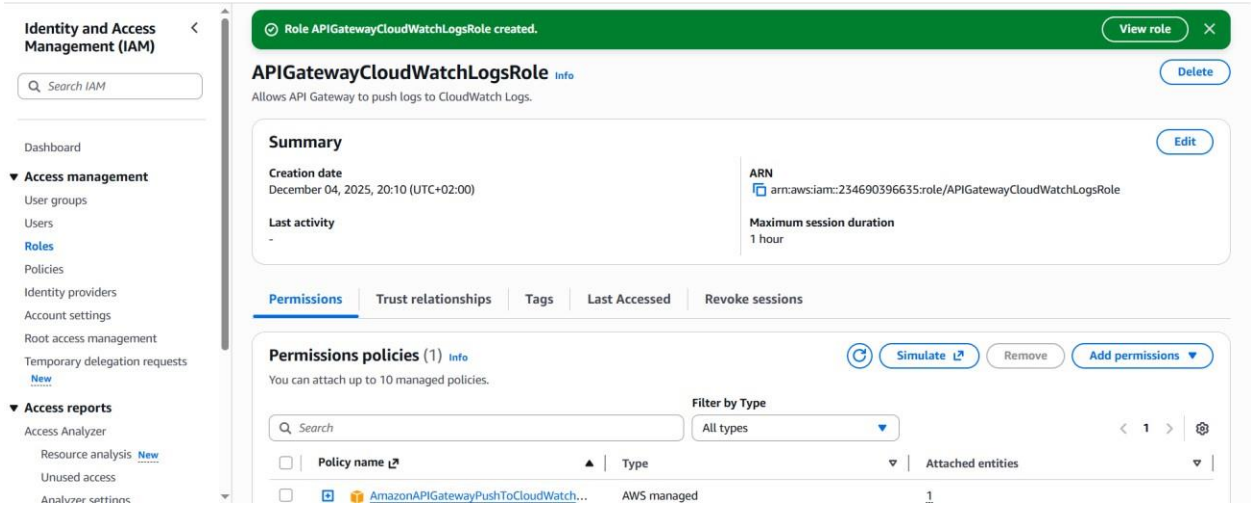
Step 6: Enforcing Security on Endpoints

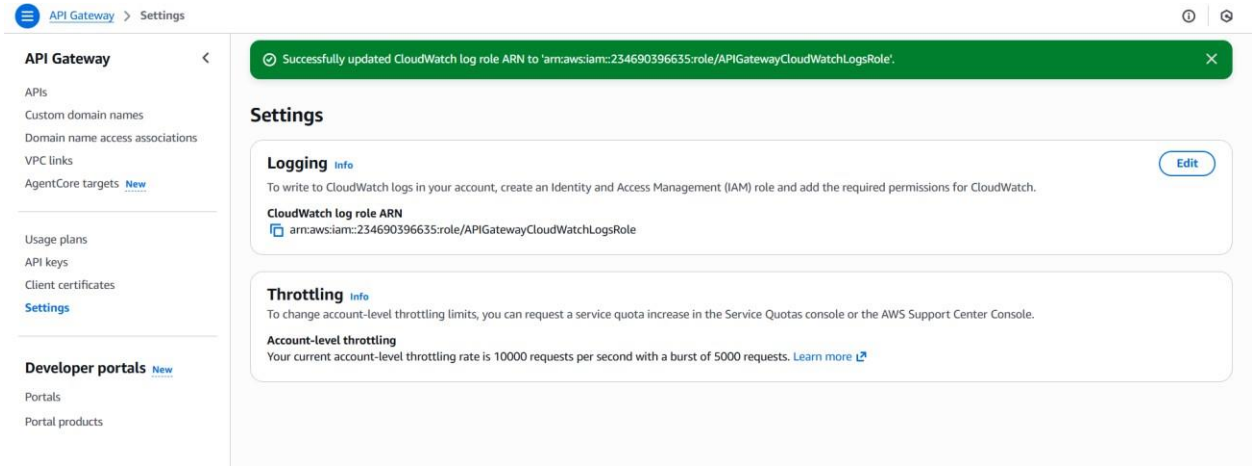
- The **Authorization** setting in the Method Request tab for all proxy methods (ANY) was changed from NONE to **`MyJWTAuthorizer`**.
- **(Crucial Check):** The **API key required** checkbox was left **unchecked**, as authentication relies solely on the JWT mechanism.
    - **Output:** All service endpoints are now secured, fulfilling the **JWT-based authentication** requirement.



Step 7: Configure Logging Permissions (IAM Prerequisite)

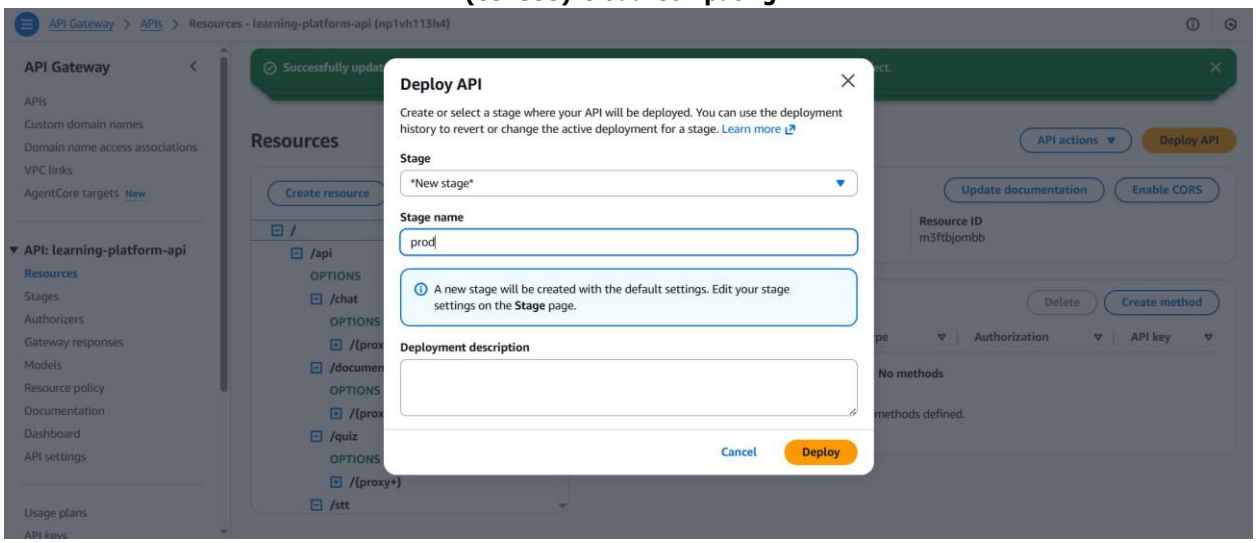
1. To grant API Gateway permission to write execution logs, a dedicated **IAM Role** named **`APIGatewayCloudWatchLogsRole`** was created.
2. The managed policy **`AmazonAPIGatewayPushToCloudWatchLogs`** was attached to this role, granting the permissions required to write logs to CloudWatch.
3. The ARN of this new role was then copied and pasted into the global **API Gateway Settings** under the **CloudWatch log role ARN** field, successfully linking the permissions to the service.
4. **Output:** This step resolved the prerequisite error for enabling logging on the deployment stage.

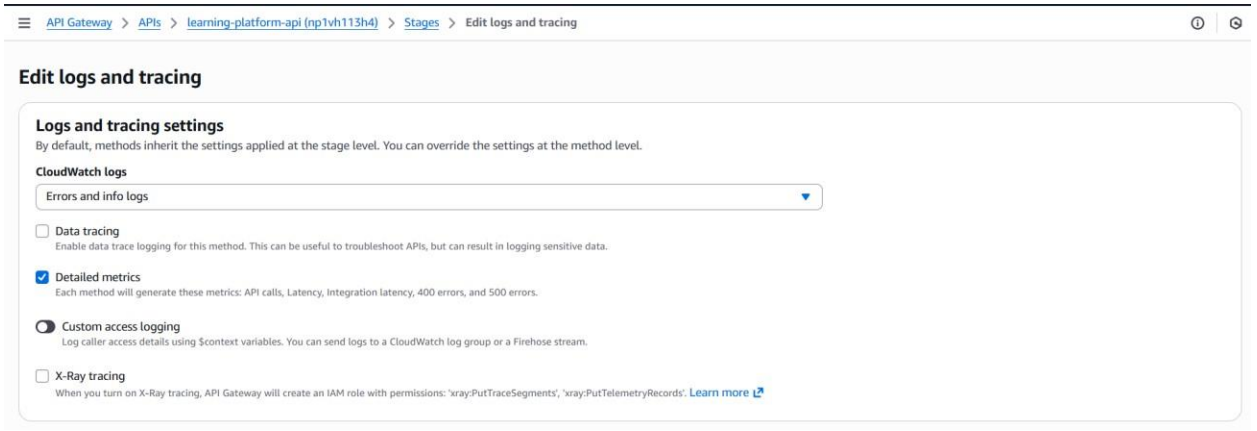


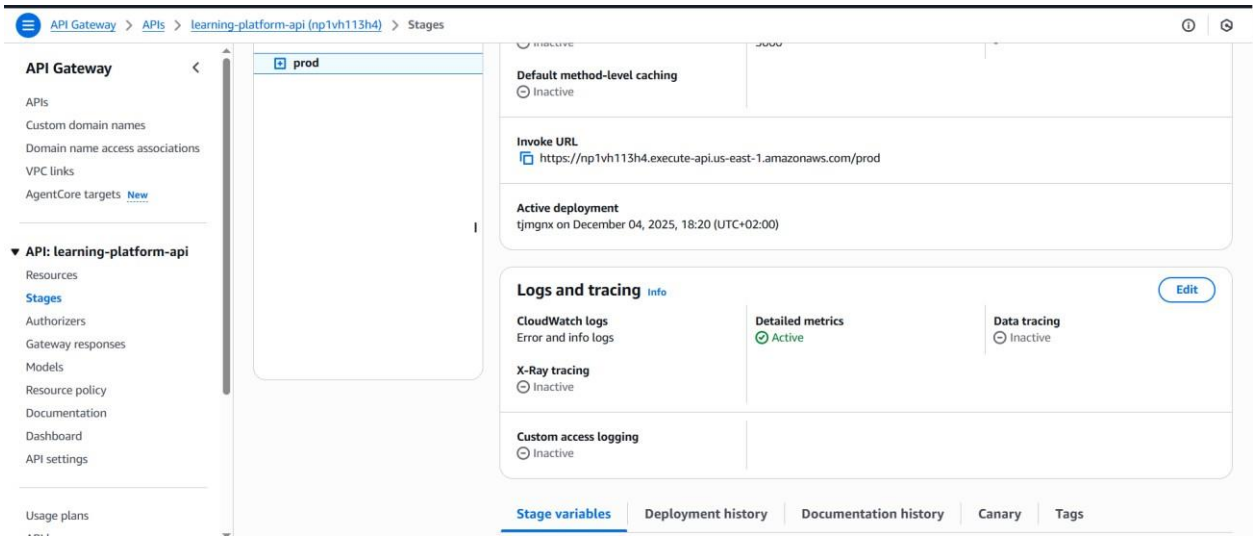


Step 8: Configuring Logging and Monitoring

- In the Stages settings for the prod stage, the **CloudWatch logs** dropdown was set to **Errors and Info logs**.
    - The **Detailed metrics** checkbox was also activated to enable granular monitoring for latency and error rates.
- **Output:** This fulfills the **Request Logging and Monitoring** requirement.







Step G: Rate Limiting and Final Deployment

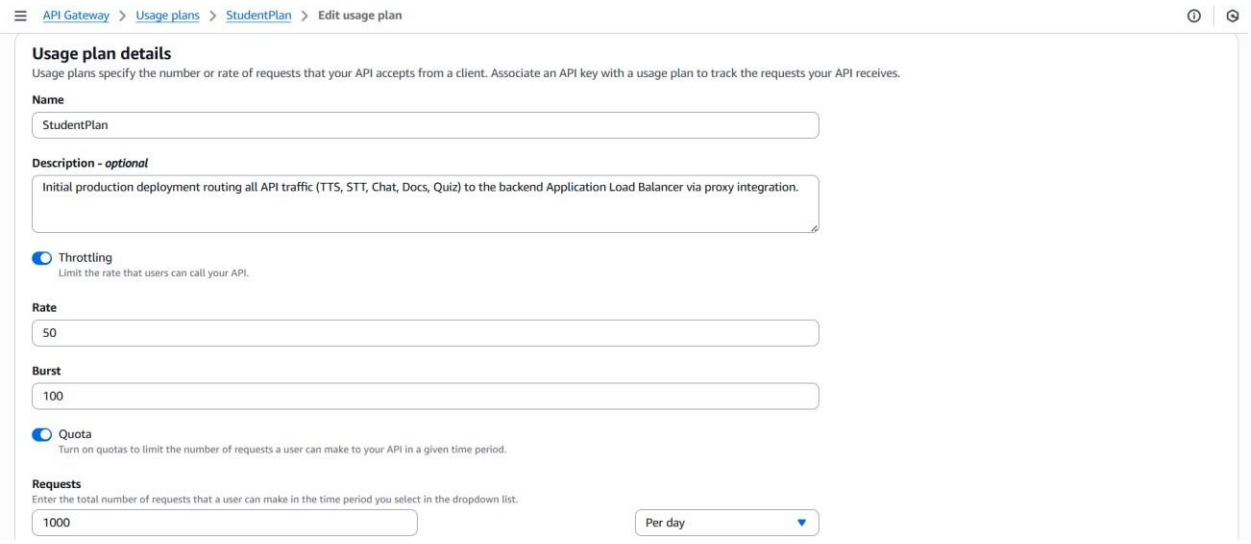
1. A **Usage Plan** (e.g., `StudentPlan`) was created with set limits for **Rate** and **Burst** and associated with the prodstage.

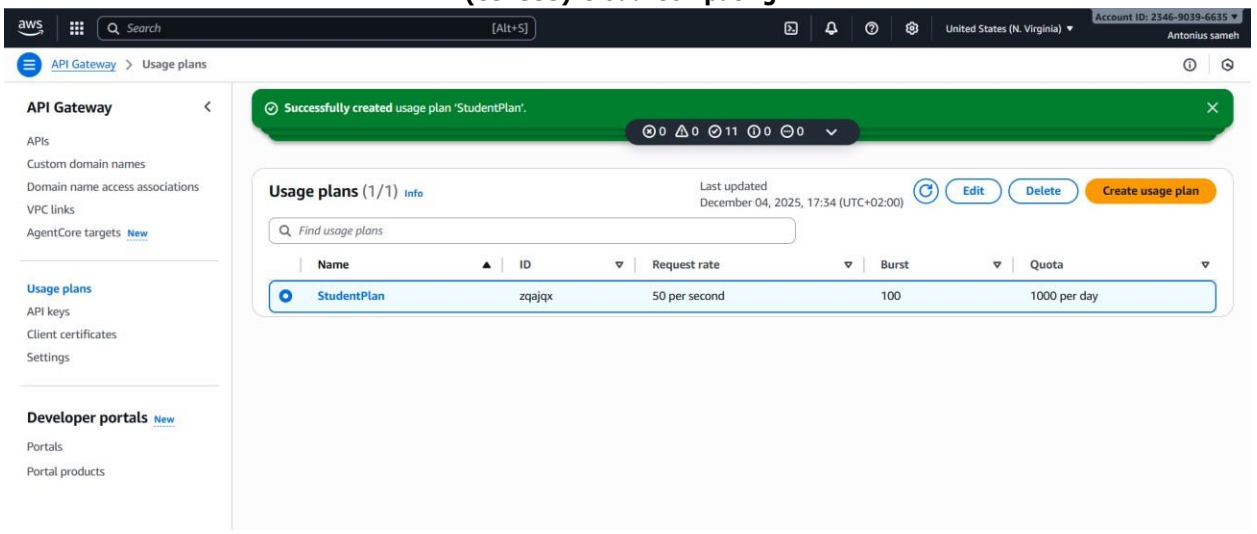
2. The plan was explicitly configured with the following constraints to limit usage:
   a. **Request Rate:** 50 requests per second.
   b. **Burst Capacity:** 100 requests.
   c. **Quota:** 1000 requests per day.

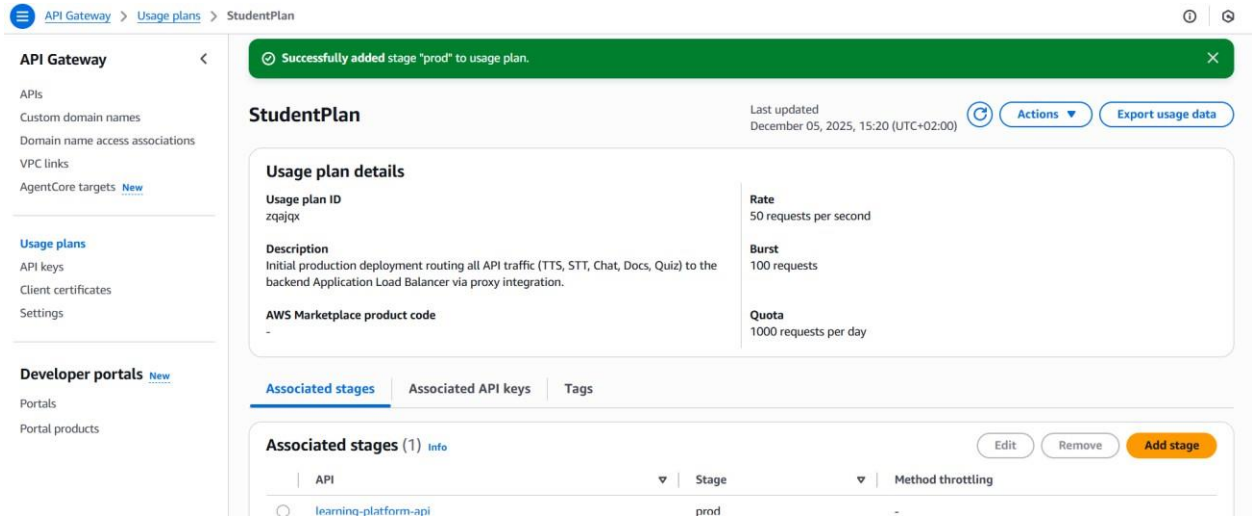
3. The `StudentPlan` was then associated with the deployed **`prod`** stage of the `learning-platform-api`, ensuring all incoming traffic adheres to these limits.

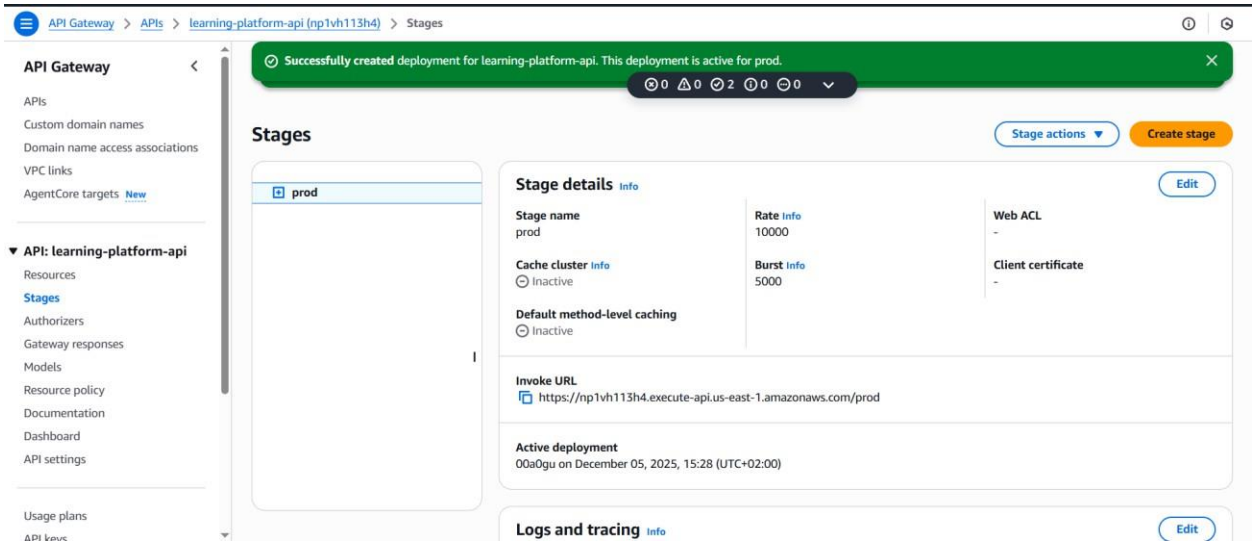
4. The entire API was finalized and deployed by clicking **"Deploy API"** and selecting the **`prod`** stage.

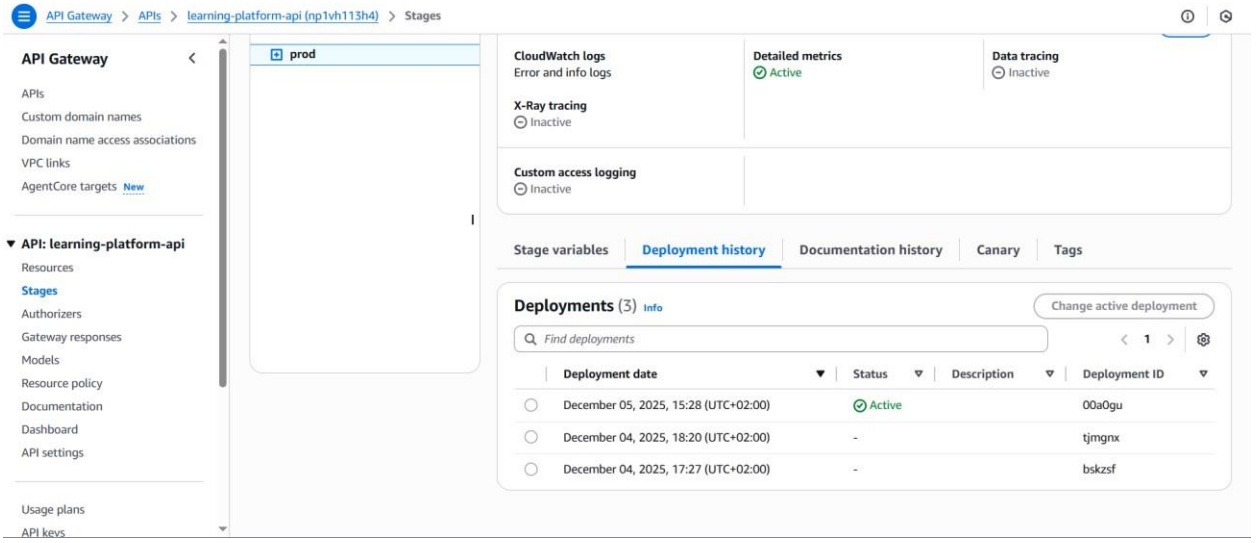
5. **Final Deliverables:** The **API Gateway configuration files** and **API documentation** were generated by exporting the OpenAPI JSON from the final deployment stage.




aws
Search
[Alt+S]
United States (N. Virginia)
Account ID: 2346-9039-6635
Antonius sameh
API Gateway > Usage plans
API Gateway
APIs
Custom domain names
Domain name access associations
VPC links
AgentCore targets New
Usage plans
API keys
Client certificates
Settings
Developer portals New
Portals
Portal products
Successfully created usage plan 'StudentPlan'.
Usage plans (1/1) Info
Last updated
December 04, 2025, 17:34 (UTC+02:00)
Edit
Delete
Create usage plan
Find usage plans
Name
ID
Request rate
Burst
Quota
StudentPlan
zqajqx
50 per second
100
1000 per day


API Gateway > Usage plans > StudentPlan
API Gateway
APIs
Custom domain names
Domain name access associations
VPC links
AgentCore targets New
Usage plans
API keys
Client certificates
Settings
Developer portals New
Portals
Portal products
Successfully added stage "prod" to usage plan.
StudentPlan
Last updated
December 05, 2025, 15:20 (UTC+02:00)
Actions
Export usage data
Usage plan details
Usage plan ID
zqajqx
Description
Initial production deployment routing all API traffic (TTS, STT, Chat, Docs, Quiz) to the backend Application Load Balancer via proxy integration.
AWS Marketplace product code
-
Rate
50 requests per second
Burst
100 requests
Quota
1000 requests per day
Associated stages
Associated API keys
Tags
Associated stages (1) Info
Edit
Remove
Add stage
API
Stage
Method throttling
learning-platform-api
prod
-


API Gateway > APIs > learning-platform-api (np1vh113h4) > Stages
API Gateway
APIs
Custom domain names
Domain name access associations
VPC links
AgentCore targets New
API: learning-platform-api
Resources
Stages
Authorizers
Gateway responses
Models
Resource policy
Documentation
Dashboard
API settings
Usage plans
Successfully created deployment for learning-platform-api. This deployment is active for prod.
Stages
Stage actions
Create stage
prod
Stage details Info
Edit
Stage name
prod
Rate Info
10000
Web ACL
-
Cache cluster Info
Inactive
Burst Info
5000
Client certificate
-
Default method-level caching
Inactive
Invoke URL
https://np1vh113h4.execute-api.us-east-1.amazonaws.com/prod
Active deployment
00a0gu on December 05, 2025, 15:28 (UTC+02:00)
Logs and tracing Info
Edit



III. API Configuration and Documentation

This requirement is met by a single generated file containing the complete technical specification of the deployment.

- **Deliverable: OpenAPI (Swagger) Specification File (JSON).**
  - **Proof of Configuration:** The file was exported using the **API Gateway Extensions** option. This ensures the output includes all underlying integration logic, such as the direct links to the **Application Load Balancer (ALB)** and the invocation setup for the **Lambda Authorizer**.


```
{
  "swagger" : "2.0", "info" : {
    "description" : "Main API Gateway for cloud-based learning platform microservices\n",
    "version" : "2025-12-04T15:27:00Z",
    "title" : "learning-platform-api"
  },
  "host" : "np1vh113h4.execute-api.us-east-1.amazonaws.com", "basePath" :
  "/prod",
  "schemes" : [ "https" ], "paths" : {
    "/api" : {
      "options" : {
        "consumes" : [ "application/json" ], "produces" : [
        "application/json" ],
```

```
        "responses" : {
          "200" : {
            "description" : "200 response",
            "schema" : {
              "$ref" : "#/definitions/Empty"
            },
            "headers" : {
              "Access-Control-Allow-Origin" : { "type" :
                "string"
              },
              "Access-Control-Allow-Methods" : { "type" :
                "string"
              },
              "Access-Control-Allow-Headers" : { "type" :
                "string"
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : { "responses" : {
            "default" : { "statusCode" :
              "200",
              "responseParameters" : {
                "method.response.header.Access-Control-Allow-Methods" :
"'DELETE,GET,HEAD,OPTIONS,PATCH,POST,PUT'",
                "method.response.header.Access-Control-Allow-Headers" : "'Content-
Type,Authorization,X-Amz-Date,X-Api-Key,X-Amz-Security-Token'",
                "method.response.header.Access-Control-Allow-Origin" : "'*'"
              }
            }
          },
          "requestTemplates" : {
            "application/json" : "{\"statusCode\": 200}"
          },
          "passthroughBehavior" : "when_no_match", "type" :
          "mock"
        }
      }
    },
      "/api/chat" : {
        "options" : {
        "consumes" : [ "application/json" ],
        "produces" : [ "application/json" ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "schema" : {
```

"$ref" : "#/definitions/Empty"

```
            },
            "headers" : {
              "Access-Control-Allow-Origin" : { "type" :
                "string"
              },
              "Access-Control-Allow-Methods" : { "type" :
                "string"
              },
              "Access-Control-Allow-Headers" : { "type" :
                "string"
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : { "responses" : {
            "default" : { "statusCode" :
              "200",
              "responseParameters" : {
                "method.response.header.Access-Control-Allow-Methods" :
'"DELETE,GET,HEAD,OPTIONS,PATCH,POST,PUT"',
                "method.response.header.Access-Control-Allow-Headers" : "'Content-
Type,Authorization,X-Amz-Date,X-Api-Key,X-Amz-Security-Token'",
                "method.response.header.Access-Control-Allow-Origin" : "'*'"
              }
            }
          },
          "requestTemplates" : {
            "application/json" : "{\"statusCode\": 200}"
          },
          "passthroughBehavior" : "when_no_match", "type" :
          "mock"
        }
      }
    },
    "/api/chat/{proxy+}" : {
      "options" : {
        "consumes" : [ "application/json" ],
        "produces" : [ "application/json" ],
        "parameters" : [ {
          "name" : "proxy",
          "in" : "path", "required"
          : true, "type" : "string"
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "schema" : {
```

```
                "$ref" : "#/definitions/Empty"
              },
              "headers" : {
                "Access-Control-Allow-Origin" : { "type" :
                  "string"
                },
                "Access-Control-Allow-Methods" : { "type" :
                  "string"
                },
                "Access-Control-Allow-Headers" : { "type" :
                  "string"
                }
              }
            }
          },
          "x-amazon-apigateway-integration" : { "responses" : {
              "default" : { "statusCode" :
                "200",
                "responseParameters" : {
                  "method.response.header.Access-Control-Allow-Methods" :
"'DELETE,GET,HEAD,OPTIONS,PATCH,POST,PUT'",
                  "method.response.header.Access-Control-Allow-Headers" : "'Content-
Type,Authorization,X-Amz-Date,X-Api-Key,X-Amz-Security-Token'",
                  "method.response.header.Access-Control-Allow-Origin" : "'*'"
                }
              }
            },
            "requestTemplates" : {
              "application/json" : "{\"statusCode\": 200}"
            },
            "passthroughBehavior" : "when_no_match", "type" :
            "mock"
          }
        },
        "x-amazon-apigateway-any-method" : { "produces" : [
          "application/json" ], "parameters" : [ {
            "name" : "proxy",
            "in" : "path", "required"
            : true, "type" : "string"
          } ],
          "responses" : { },
          "x-amazon-apigateway-integration" : {
            "uri" : "http://dev-public-alb-954135140.us-east- 1.elb.amazonaws.com/api/chat/{proxy}",
            "httpMethod" : "ANY",
            "responses" : {
```

```
            "default" : { "statusCode" :
              "200"
            }
          },
          "requestParameters" : {
            "integration.request.path.proxy" : "method.request.path.proxy"
          },
          "passthroughBehavior" : "when_no_templates",
          "type" : "http"
        }
      }
    },
    "/api/documents"  :  {
      "options" : {
        "consumes"  :  [  "application/json"  ],
        "produces"  :  [  "application/json"  ],
        "responses" : {
          "200" : {
            "description"  :  "200  response",
            "schema" : {
              "$ref" : "#/definitions/Empty"
            },
            "headers" : {
              "Access-Control-Allow-Origin" : { "type" :
                "string"
              },
              "Access-Control-Allow-Methods" : { "type" :
                "string"
              },
              "Access-Control-Allow-Headers" : { "type" :
                "string"
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : { "responses" : {
            "default"  : { "statusCode" :
              "200",
              "responseParameters" : {
                "method.response.header.Access-Control-Allow-Methods" :
'"DELETE,GET,HEAD,OPTIONS,PATCH,POST,PUT"',
                "method.response.header.Access-Control-Allow-Headers" : '"Content-
Type,Authorization,X-Amz-Date,X-Api-Key,X-Amz-Security-Token"',
                "method.response.header.Access-Control-Allow-Origin"  :  '"*"'
              }
            }
          },
```

```
"requestTemplates" : {
```

```
            "application/json" : "{\"statusCode\": 200}"
          },
          "passthroughBehavior" : "when_no_match", "type" :
          "mock"
        }
      }
    },
    "/api/documents/{proxy+}"   :   {
      "options" : {
        "consumes"   :   [   "application/json"   ],
        "produces"   :   [   "application/json"   ],
        "parameters" : [ {
          "name" : "proxy",
          "in" : "path", "required"
          : true, "type" : "string"
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "schema" : {
              "$ref" : "#/definitions/Empty"
            },
            "headers" : {
              "Access-Control-Allow-Origin" : { "type" :
                "string"
              },
              "Access-Control-Allow-Methods" : { "type" :
                "string"
              },
              "Access-Control-Allow-Headers" : { "type" :
                "string"
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : { "responses" : {
            "default"  : { "statusCode" :
              "200",
              "responseParameters" : {
                "method.response.header.Access-Control-Allow-Methods" :
"'DELETE,GET,HEAD,OPTIONS,PATCH,POST,PUT'",
                "method.response.header.Access-Control-Allow-Headers" : "'Content-
Type,Authorization,X-Amz-Date,X-Api-Key,X-Amz-Security-Token'",
                "method.response.header.Access-Control-Allow-Origin" : "'*'"
              }
            }
          },
```

```
            "requestTemplates" : {
              "application/json" : "{\"statusCode\": 200}"
            },
            "passthroughBehavior" : "when_no_match", "type" :
            "mock"
          }
        },
        "x-amazon-apigateway-any-method" : { "produces" : [
          "application/json" ], "parameters" : [ {
            "name" : "proxy",
            "in" : "path", "required"
            : true, "type" : "string"
          } ],
          "responses" : { },
          "x-amazon-apigateway-integration" : {
            "uri" : "http://dev-public-alb-954135140.us-east- 1.elb.amazonaws.com/api/documents/{proxy}",
            "httpMethod" : "ANY",
            "responses" : {
              "default" : { "statusCode" :
                "200"
              }
            },
            "requestParameters" : {
              "integration.request.path.proxy" : "method.request.path.proxy"
            },
            "passthroughBehavior" : "when_no_templates",
            "type" : "http"
          }
        }
      },
        "/api/quiz" : {
          "options" : {
        "consumes"    :    [    "application/json"    ],
        "produces"    :    [    "application/json"    ],
        "responses" : {
          "200" : {
            "description"    :    "200    response",
            "schema" : {
              "$ref" : "#/definitions/Empty"
            },
            "headers" : {
              "Access-Control-Allow-Origin" : { "type" :
                "string"
              },
              "Access-Control-Allow-Methods" : { "type" :
                "string"
```

```
                },
                "Access-Control-Allow-Headers" : { "type" :
                  "string"
                }
              }
            }
          },
          "x-amazon-apigateway-integration" : { "responses" : {
              "default" : { "statusCode" :
                "200",
                "responseParameters" : {
                  "method.response.header.Access-Control-Allow-Methods" :
"'DELETE,GET,HEAD,OPTIONS,PATCH,POST,PUT'",
                  "method.response.header.Access-Control-Allow-Headers" : "'Content-
Type,Authorization,X-Amz-Date,X-Api-Key,X-Amz-Security-Token'",
                  "method.response.header.Access-Control-Allow-Origin" : "'*'"
                }
              }
            },
            "requestTemplates" : {
              "application/json" : "{\"statusCode\": 200}"
            },
            "passthroughBehavior" : "when_no_match", "type" :
            "mock"
          }
        }
      },
      "/api/quiz/{proxy+}"    :    {
        "options" : {
          "consumes"    :    [    "application/json"    ],
          "produces"    :    [    "application/json"    ],
          "parameters" : [ {
            "name" : "proxy",
            "in" : "path", "required"
            : true, "type" : "string"
          } ],
          "responses" : {
            "200" : {
              "description" : "200 response",
              "schema" : {
                "$ref" : "#/definitions/Empty"
              },
              "headers" : {
                "Access-Control-Allow-Origin" : { "type" :
                  "string"
                },
                "Access-Control-Allow-Methods" : {
```

```
                    "type" : "string"
                  },
                  "Access-Control-Allow-Headers" : { "type" :
                    "string"
                  }
                }
              }
            },
            "x-amazon-apigateway-integration" : { "responses" : {
                "default" : { "statusCode" :
                  "200",
                  "responseParameters" : {
                    "method.response.header.Access-Control-Allow-Methods" :
"'DELETE,GET,HEAD,OPTIONS,PATCH,POST,PUT'",
                    "method.response.header.Access-Control-Allow-Headers" : "'Content-
Type,Authorization,X-Amz-Date,X-Api-Key,X-Amz-Security-Token'",
                    "method.response.header.Access-Control-Allow-Origin" : "'*'"
                  }
                }
              },
              "requestTemplates" : {
                "application/json" : "{\"statusCode\": 200}"
              },
              "passthroughBehavior" : "when_no_match", "type" :
              "mock"
            }
          },
          "x-amazon-apigateway-any-method" : { "produces" : [
            "application/json" ], "parameters" : [ {
              "name" : "proxy",
              "in" : "path", "required"
              : true, "type" : "string"
            } ],
            "responses" : { },
            "x-amazon-apigateway-integration" : {
              "uri" : "http://dev-public-alb-954135140.us-east- 1.elb.amazonaws.com/api/quiz/{proxy}",
              "httpMethod" : "ANY",
              "responses" : {
                "default" : { "statusCode" :
                  "200"
                }
              },
              "requestParameters" : {
                "integration.request.path.proxy" : "method.request.path.proxy"
              },
```

```
            "passthroughBehavior" : "when_no_templates",
            "type" : "http"
          }
        }
      },
      "/api/stt" : {
        "options" : {
          "consumes"  :  [  "application/json"  ],
          "produces"  :  [  "application/json"  ],
          "responses" : {
            "200" : {
              "description"  :  "200  response",
              "schema" : {
                "$ref" : "#/definitions/Empty"
              },
              "headers" : {
                "Access-Control-Allow-Origin" : { "type" :
                  "string"
                },
                "Access-Control-Allow-Methods" : { "type" :
                  "string"
                },
                "Access-Control-Allow-Headers" : { "type" :
                  "string"
                }
              }
            }
          },
          "x-amazon-apigateway-integration" : { "responses" : {
              "default" : { "statusCode" :
                "200",
                "responseParameters" : {
                  "method.response.header.Access-Control-Allow-Methods" :
"'DELETE,GET,HEAD,OPTIONS,PATCH,POST,PUT'",
                  "method.response.header.Access-Control-Allow-Headers" : "'Content-
Type,Authorization,X-Amz-Date,X-Api-Key,X-Amz-Security-Token'",
                  "method.response.header.Access-Control-Allow-Origin" : "'*'"
                }
              }
            },
            "requestTemplates" : {
              "application/json" : "{\"statusCode\": 200}"
            },
            "passthroughBehavior" : "when_no_match", "type" :
            "mock"
          }
        }
```

```
},
```

```
    "/api/stt/{proxy+}"    :    {
      "options" : {
        "consumes"    :    [    "application/json"    ],
        "produces"    :    [    "application/json"    ],
        "parameters" : [ {
          "name" : "proxy",
          "in" : "path", "required"
          : true, "type" : "string"
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "schema" : {
              "$ref" : "#/definitions/Empty"
            },
            "headers" : {
              "Access-Control-Allow-Origin" : { "type" :
                "string"
              },
              "Access-Control-Allow-Methods" : { "type" :
                "string"
              },
              "Access-Control-Allow-Headers" : { "type" :
                "string"
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : { "responses" : {
            "default" : { "statusCode" :
              "200",
              "responseParameters" : {
                "method.response.header.Access-Control-Allow-Methods" :
"'DELETE,GET,HEAD,OPTIONS,PATCH,POST,PUT'",
                "method.response.header.Access-Control-Allow-Headers" : "'Content-
Type,Authorization,X-Amz-Date,X-Api-Key,X-Amz-Security-Token'",
                "method.response.header.Access-Control-Allow-Origin" : "'*'"
              }
            }
          },
          "requestTemplates" : {
            "application/json" : "{\"statusCode\": 200}"
          },
          "passthroughBehavior" : "when_no_match", "type" :
          "mock"
        }
      },
```

```
      "x-amazon-apigateway-any-method" : { "produces" : [
        "application/json" ], "parameters" : [ {
          "name" : "proxy",
          "in" : "path", "required"
          : true, "type" : "string"
        } ],
        "responses" : { },
        "x-amazon-apigateway-integration" : {
          "uri" : "http://dev-public-alb-954135140.us-east- 1.elb.amazonaws.com/api/stt/{proxy}",
          "httpMethod" : "ANY",
          "responses" : {
            "default" : { "statusCode" :
              "200"
            }
          },
          "requestParameters" : {
            "integration.request.path.proxy" : "method.request.path.proxy"
          },
          "passthroughBehavior" : "when_no_templates",
          "type" : "http"
        }
      }
    },
    "/api/tts" : {
      "options" : {
        "consumes" : [ "application/json" ],
        "produces" : [ "application/json" ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "schema" : {
              "$ref" : "#/definitions/Empty"
            },
            "headers" : {
              "Access-Control-Allow-Origin" : { "type" :
                "string"
              },
              "Access-Control-Allow-Methods" : { "type" :
                "string"
              },
              "Access-Control-Allow-Headers" : { "type" :
                "string"
              }
            }
          }
        },
```

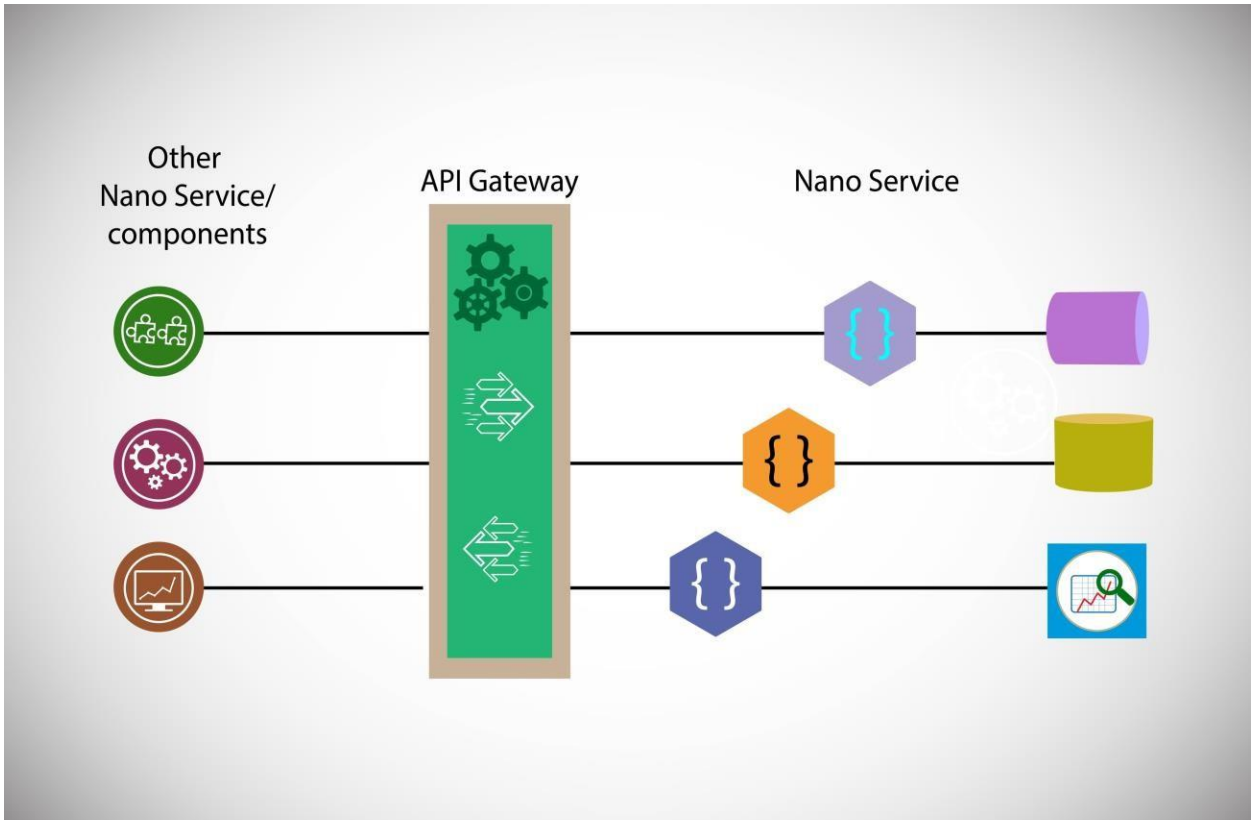
```
        "x-amazon-apigateway-integration" : { "responses" : {
            "default" : { "statusCode" :
              "200",
              "responseParameters" : {
                "method.response.header.Access-Control-Allow-Methods" :
'"DELETE,GET,HEAD,OPTIONS,PATCH,POST,PUT"',
                "method.response.header.Access-Control-Allow-Headers" : '"Content-
Type,Authorization,X-Amz-Date,X-Api-Key,X-Amz-Security-Token"',
                "method.response.header.Access-Control-Allow-Origin" : '"*"'
              }
            }
          },
          "requestTemplates" : {
            "application/json" : "{\"statusCode\": 200}"
          },
          "passthroughBehavior" : "when_no_match", "type" :
          "mock"
        }
      }
    },
    "/api/tts/{proxy+}"    :    {
      "options" : {
        "consumes"    :    [    "application/json"    ],
        "produces"    :    [    "application/json"    ],
        "parameters" : [ {
          "name" : "proxy",
          "in" : "path", "required"
          : true, "type" : "string"
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "schema" : {
              "$ref" : "#/definitions/Empty"
            },
            "headers" : {
              "Access-Control-Allow-Origin" : { "type" :
                "string"
              },
              "Access-Control-Allow-Methods" : { "type" :
                "string"
              },
              "Access-Control-Allow-Headers" : { "type" :
                "string"
              }
            }
          }
```

```
        },
        "x-amazon-apigateway-integration" : { "responses" : {
            "default" : { "statusCode" :
              "200",
              "responseParameters" : {
                "method.response.header.Access-Control-Allow-Methods" :
'"DELETE,GET,HEAD,OPTIONS,PATCH,POST,PUT"',
                "method.response.header.Access-Control-Allow-Headers" : "'Content-
Type,Authorization,X-Amz-Date,X-Api-Key,X-Amz-Security-Token'",
                "method.response.header.Access-Control-Allow-Origin" : "'*'"
              }
            }
          },
          "requestTemplates" : {
            "application/json" : "{\"statusCode\": 200}"
          },
          "passthroughBehavior" : "when_no_match", "type" :
          "mock"
        }
      },
      "x-amazon-apigateway-any-method" : { "produces" : [
        "application/json" ], "parameters" : [ {
          "name" : "proxy",
          "in" : "path", "required"
          : true, "type" : "string"
        } ],
        "responses" : { },
        "x-amazon-apigateway-integration" : {
          "uri" : "http://dev-public-alb-954135140.us-east- 1.elb.amazonaws.com/api/tts/{proxy}",
          "httpMethod" : "ANY",
          "responses" : {
            "default" : { "statusCode" :
              "200"
            }
          },
          "requestParameters" : {
            "integration.request.path.proxy" : "method.request.path.proxy"
          },
          "passthroughBehavior" : "when_no_templates",
          "type" : "http"
        }
      }
    }
  },
  "definitions" : {
```

```
    "Empty" : {
      "type" : "object", "title" :
      "Empty Schema"
    }
  },
  "x-amazon-apigateway-security-policy" : "SecurityPolicy_TLS13_1_3_2025_09", "x-
  amazon-apigateway-endpoint-access-mode" : "BASIC"
}
```

VI. Authentication Flow Diagrams

- **Deliverable: Flowchart or Detailed Narrative.**
  - **Proof of Security Mechanism:** This document outlines the security architecture where the **Lambda Authorizer** intercepts the request and validates the JSON Web Token (JWT) provided in the `Authorization` header, generating an **IAM Allow Policy** as the prerequisite for backend access.



V. Final Deployment Artifact

- **Production Endpoint (Invoke URL):** https://np1vh113h4.execute-api.us-east-1.amazonaws.com/prod
- *This URL acts as the live entry point for all client requests, verifying that the API Gateway is successfully deployed and accessible on the internet.*

5. Microservices Specifications

5.1 Text-to-Speech (TTS) Service

Overview of the TTS Microservice

Goal: Build a scalable Text-to-Speech service that converts text into audio, stores it, and integrates with Kafka for event streaming.

Key Features

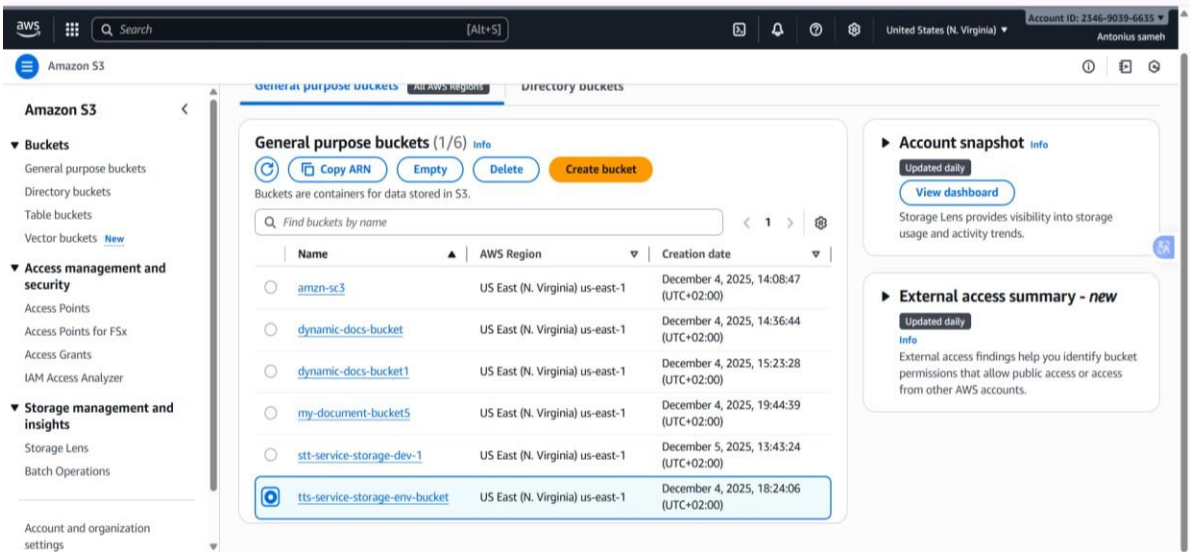
- Generate natural speech (MP3/WAV/OGG)
- Multi-language support
- REST APIs for synthesize / retrieve / delete
- Kafka event-driven pipeline
- Audio storage isolation using S3
- Containerized + Fully deployed on AWS

Step 1: Infrastructure Setup (S3 & Networking)

First, you need to set up the **storage and isolation** environment on AWS.

- C**reate S3 Bucket**: Provision a dedicated S3 bucket named **tts- service-storage-env-bucket.**
- **Purpose**: This will store all generated audio files to ensure storage isolation.
- **Networking**: Ensure the service runs within the VPC to communicate with the Kafka cluster and other services

Configure the S3 Bucket



Step 2: Microservice Development (Python & Logic)

Develop the core logic using the specified tech stack.

- **Language & Environment:** Use **Python 3.11** as the base runtime.
- **TTS Engine Integration:** Implement the speech synthesis logic using the AWS

Polly SDK, gTTS, or Coqui TTS.

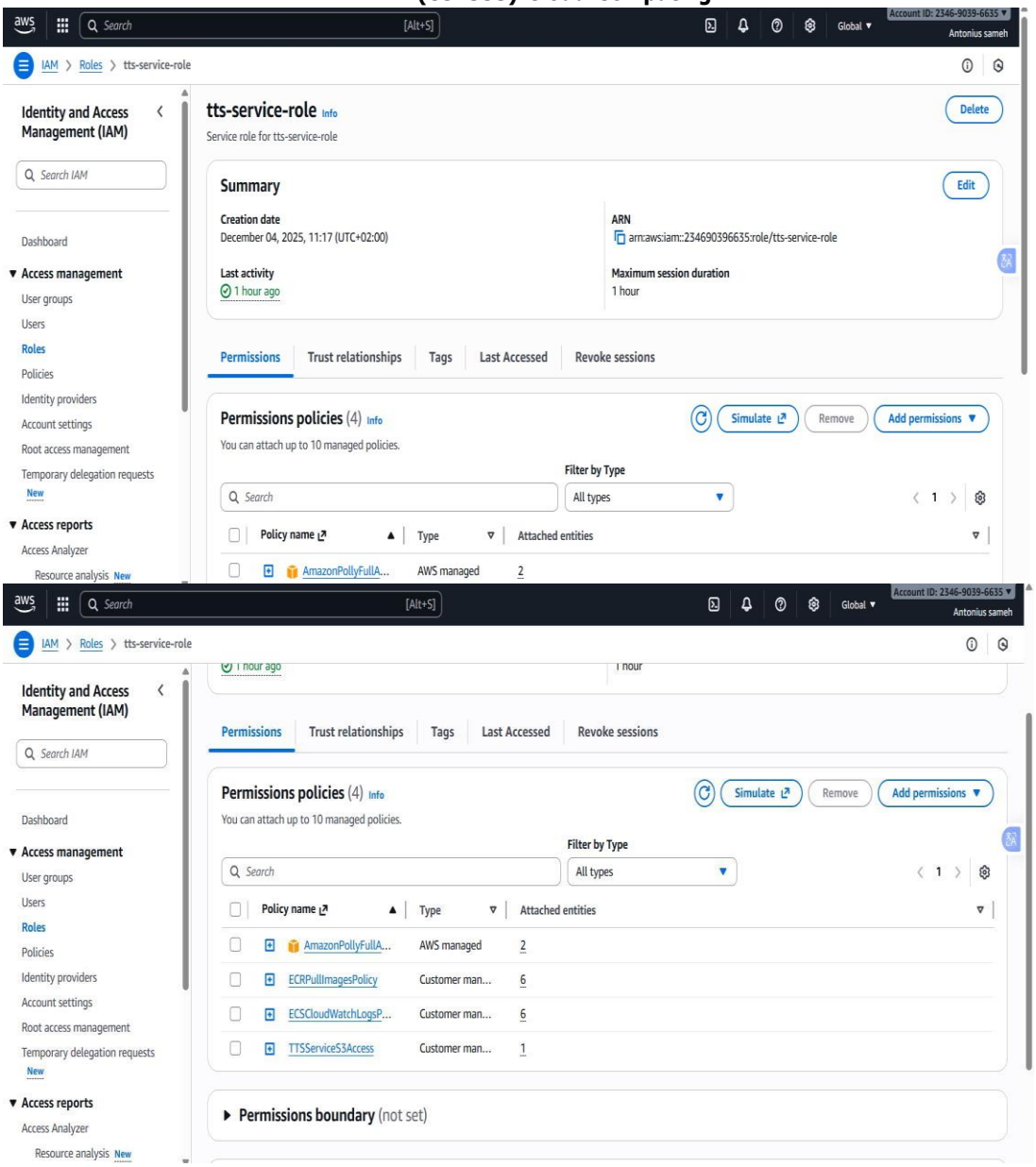
Prerequisite: IAM Permissions

- Before writing the code, the identity (Role) running your Docker container must have permissions to access Polly and S3.

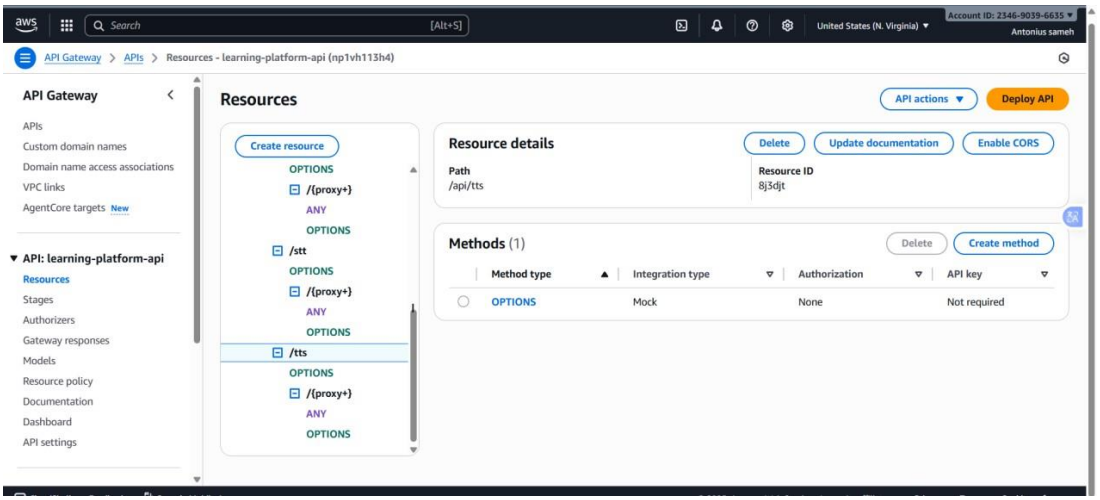
- **Policy Required**: **AmazonPollyFullAccess** and **AmazonS3FullAccess** (to save the file).

- **API Implementation**: Develop the following **REST** endpoints:
- POST /api/tts/synthesize: To trigger speech generation.
- GET /api/tts/audio/{id}: To retrieve the file.
- DELETE /api/tts/audio/{id}: To remove audio files

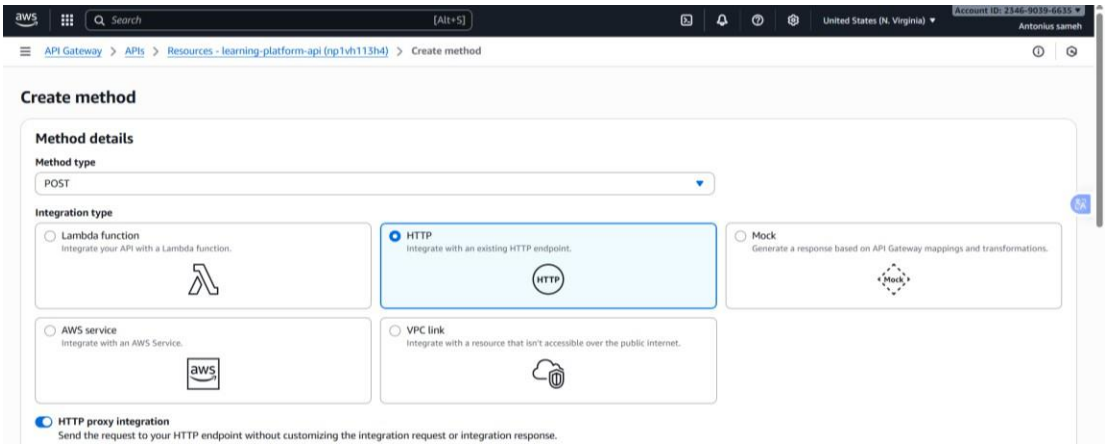
IAM Role

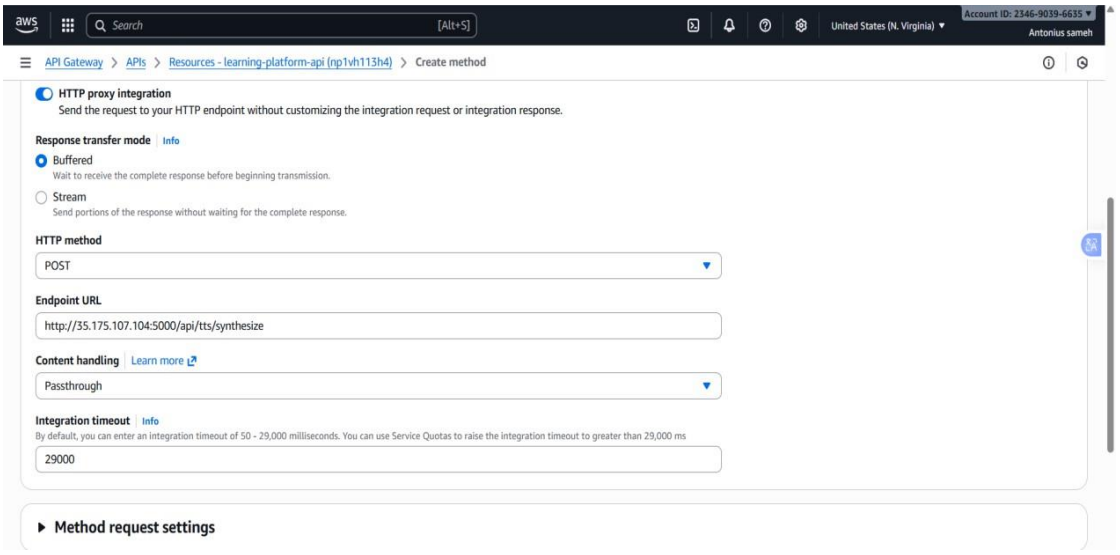

Search [Alt+S]
Global
Account ID: 2346-9039-6635
Antonius sameh
IAM > Roles > tts-service-role
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Root access management
Temporary delegation requests
New
Access reports
Access Analyzer
Resource analysis New
tts-service-role Info
Service role for tts-service-role
Delete
Summary
Edit
Creation date
December 04, 2025, 11:17 (UTC+02:00)
ARN
arn:aws:iam::234690396635:role/tts-service-role
Last activity
1 hour ago
Maximum session duration
1 hour
Permissions
Trust relationships
Tags
Last Accessed
Revoke sessions
Permissions policies (4) Info
You can attach up to 10 managed policies.
Simulate
Remove
Add permissions
Filter by Type
Search
All types
Policy name
Type
Attached entities
AmazonPollyFullA...
AWS managed
2
ECRPullImagesPolicy
Customer man...
6
ECSCloudWatchLogsP...
Customer man...
6
TTSServiceS3Access
Customer man...
1
Permissions boundary (not set)

APIs

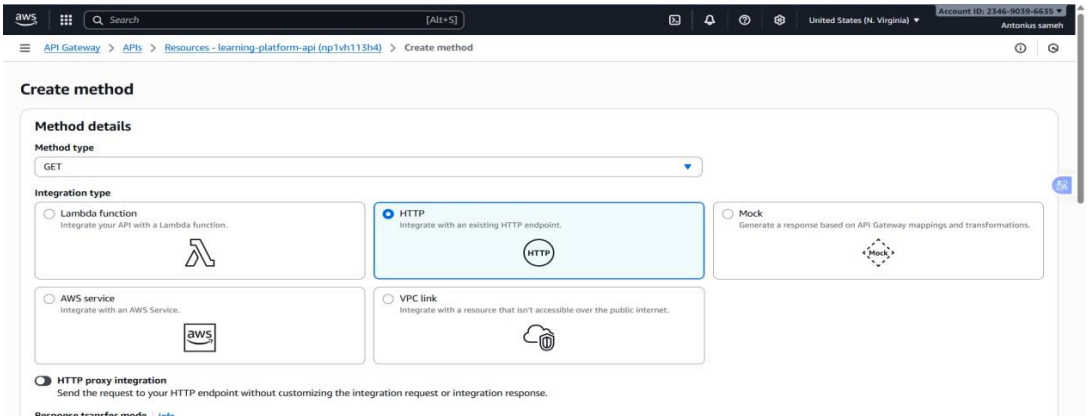

Search [Alt+S]
United States (N. Virginia)
Account ID: 2346-9039-6635
Antonius sameh
API Gateway > APIs > Resources - learning-platform-api (np1vh113h4)
API Gateway
APIs
Custom domain names
Domain name access associations
VPC links
AgentCore targets New
API: learning-platform-api
Resources
Stages
Authorizers
Gateway responses
Models
Resource policy
Documentation
Dashboard
API settings
Resources
API actions
Deploy API
Create resource
OPTIONS
/{proxy+}
ANY
OPTIONS
/stt
OPTIONS
/{proxy+}
ANY
OPTIONS
/tts
OPTIONS
/{proxy+}
ANY
OPTIONS
Resource details
Delete
Update documentation
Enable CORS
Path
/api/tts
Resource ID
8j3djt
Methods (1)
Delete
Create method
Method type
Integration type
Authorization
API key
OPTIONS
Mock
None
Not required

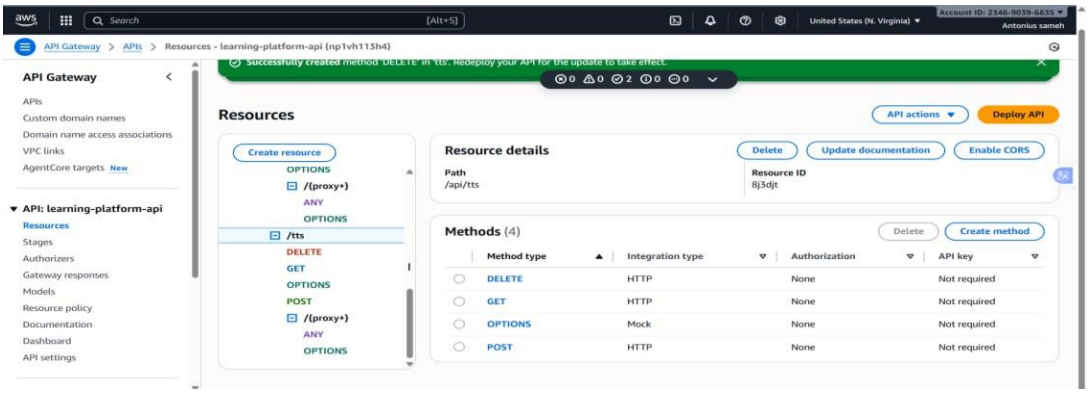
TTS-POST

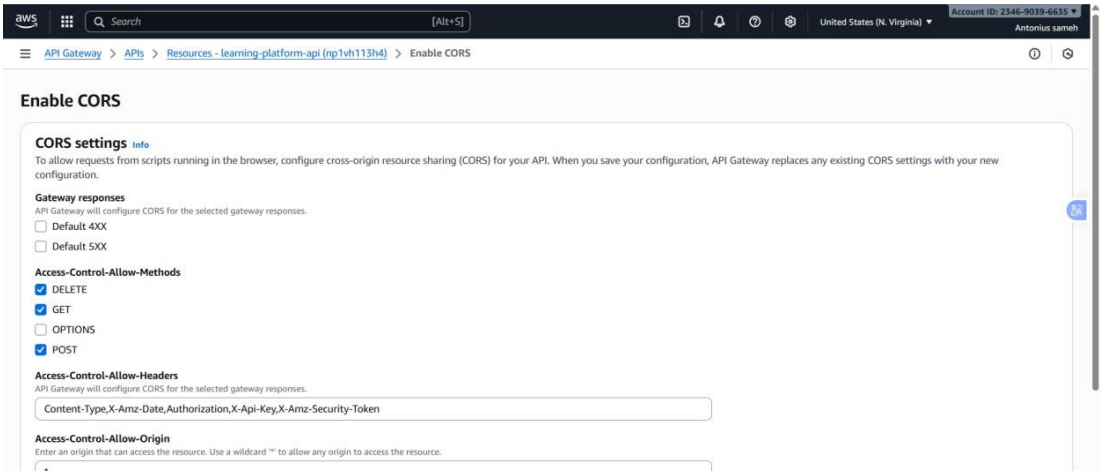




TTS-GET







Build Docker Image

Use Python 3.11 base image

Install libraries:

boto3 (Polly/S3)

Kafka client (kafka-python)

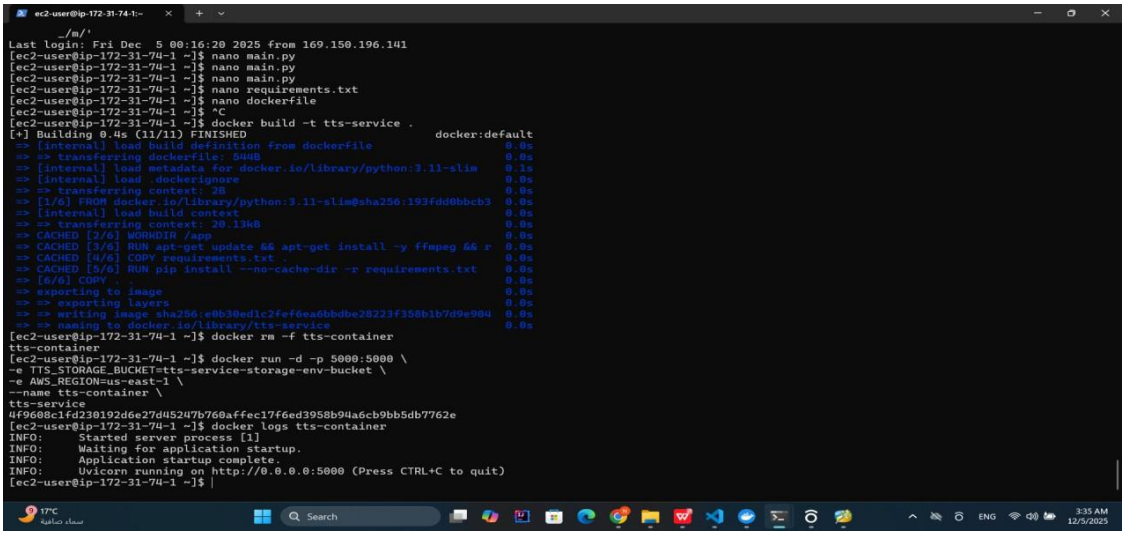
Optional: gTTS

Expose port 5000

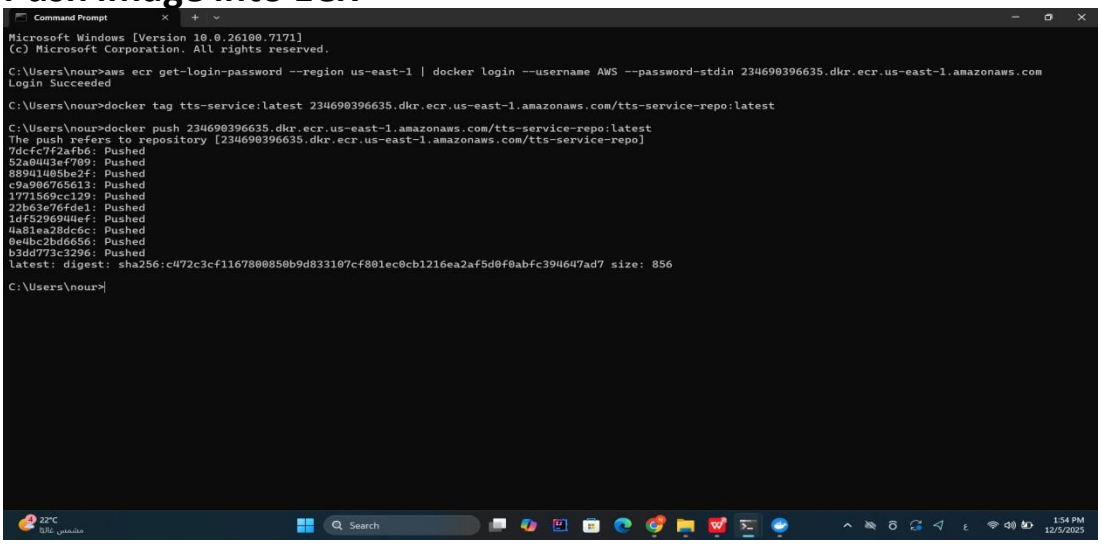
Push the Docker image to ECR (tts-service-repo) using:

docker build -t tts-service .

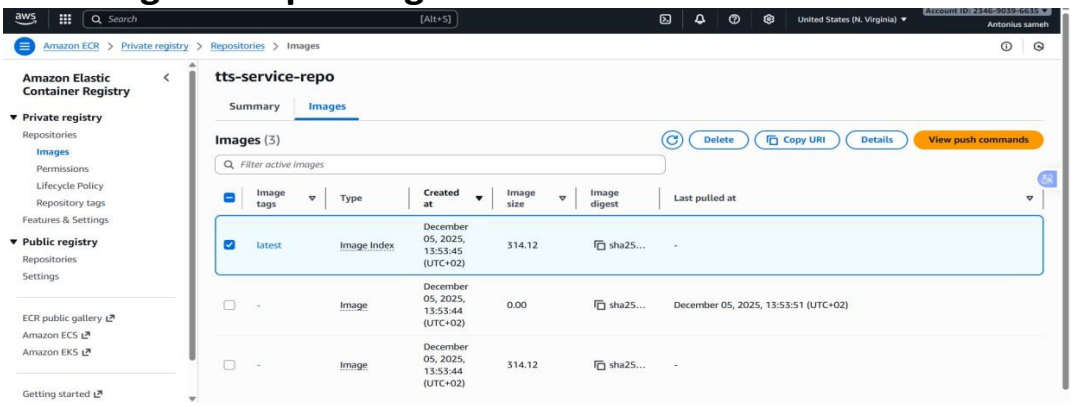
docker push <account>.dkr.ecr.<region>.amazonaws.com/tts-service-repo



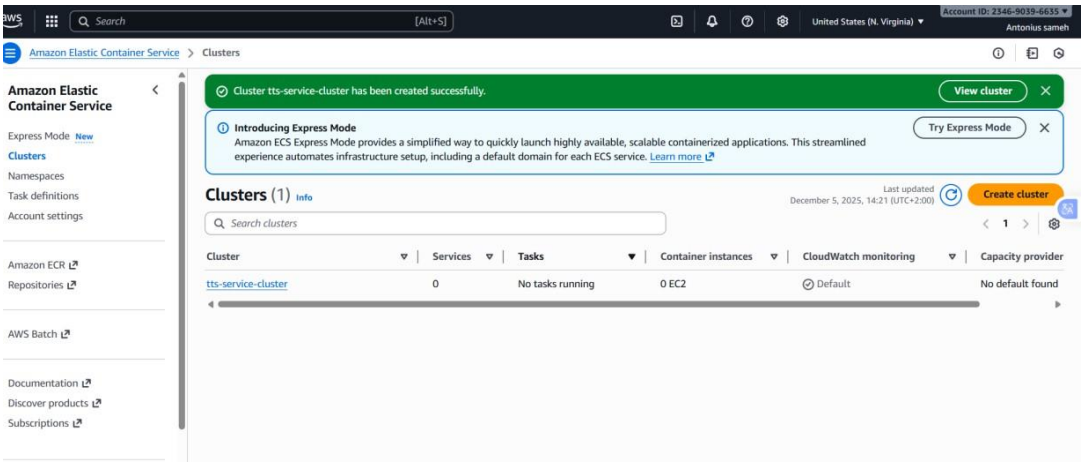
Push image into ECR



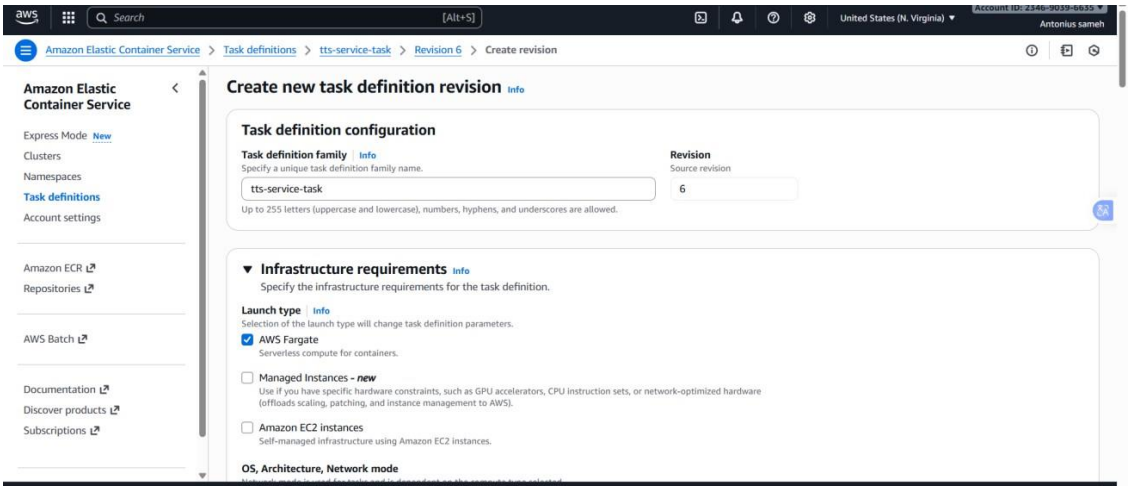
TTS image after pushing it into ECR

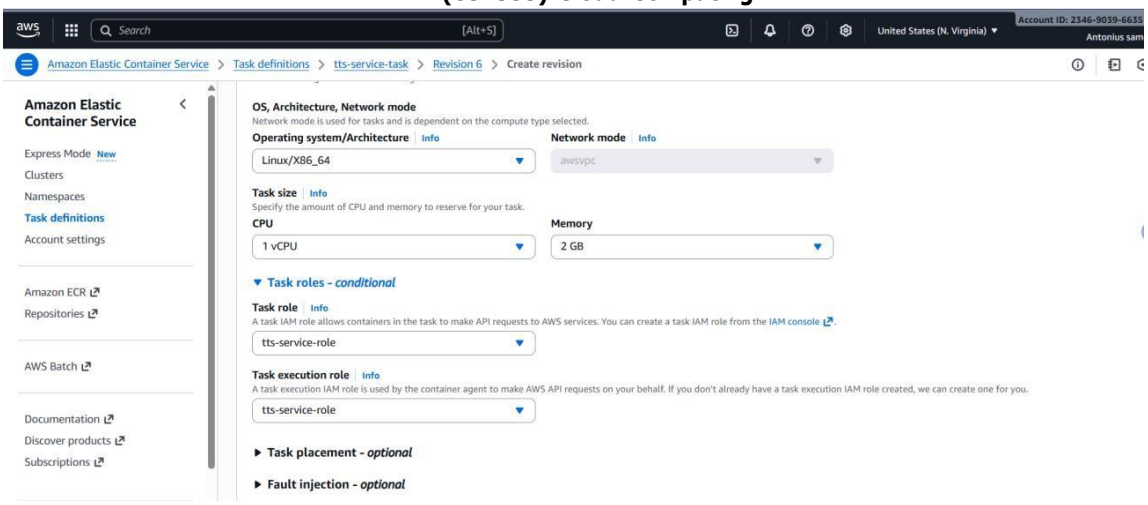


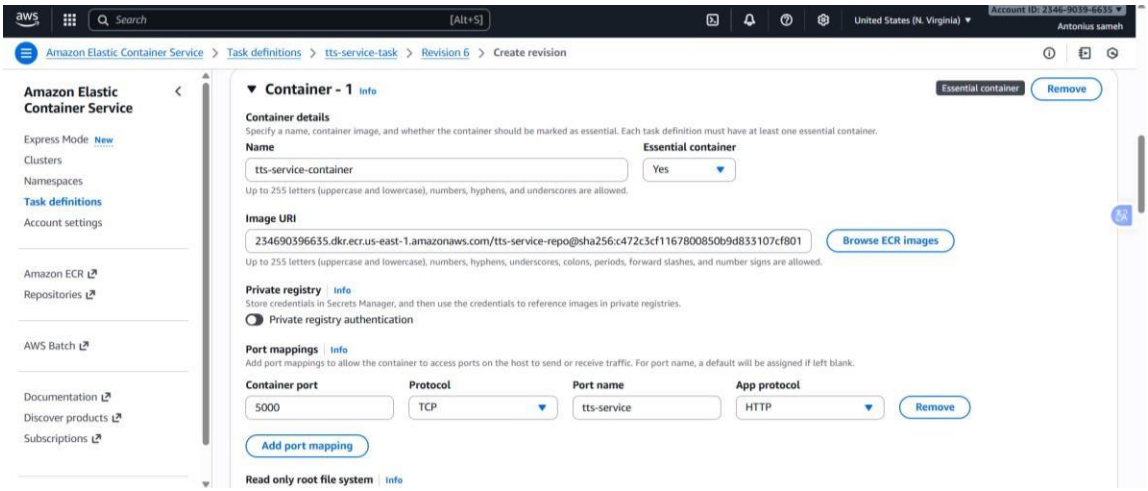
Create ECS Cluster

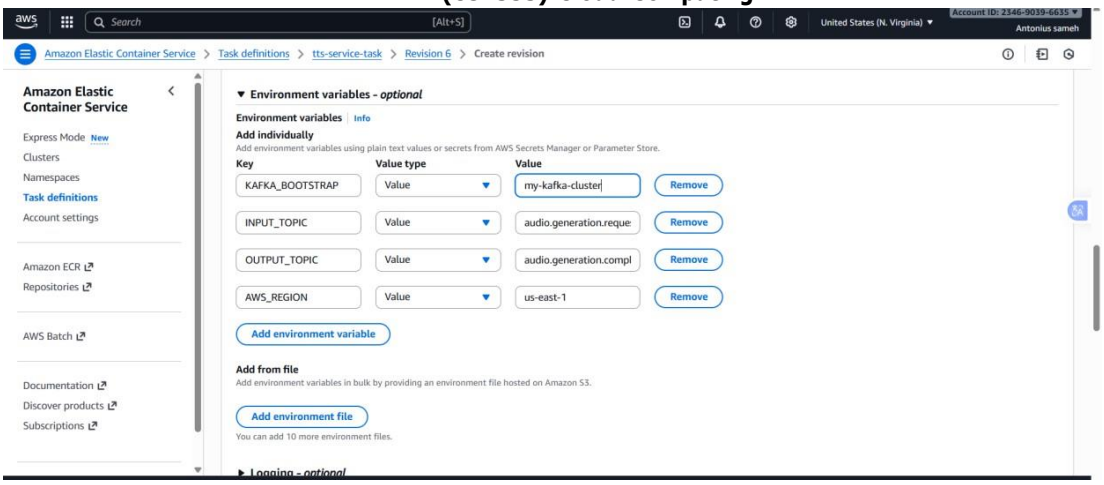


Task Definetion




Search [Alt+S]
United States (N. Virginia)
Amazon Elastic Container Service > Task definitions > tts-service-task > Revision 6 > Create revision
Amazon Elastic Container Service
Express Mode New
Clusters
Namespaces
Task definitions
Account settings
Amazon ECR
Repositories
AWS Batch
Documentation
Discover products
Subscriptions
OS, Architecture, Network mode
Network mode is used for tasks and is dependent on the compute type selected.
Operating system/Architecture Info
Linux/X86_64
Network mode Info
awsvpc
Task size Info
Specify the amount of CPU and memory to reserve for your task.
CPU
1 vCPU
Memory
2 GB
Task roles - conditional
Task role Info
A task IAM role allows containers in the task to make API requests to AWS services. You can create a task IAM role from the IAM console.
tts-service-role
Task execution role Info
A task execution IAM role is used by the container agent to make AWS API requests on your behalf. If you don't already have a task execution IAM role created, we can create one for you.
tts-service-role
Task placement - optional
Fault injection - optional

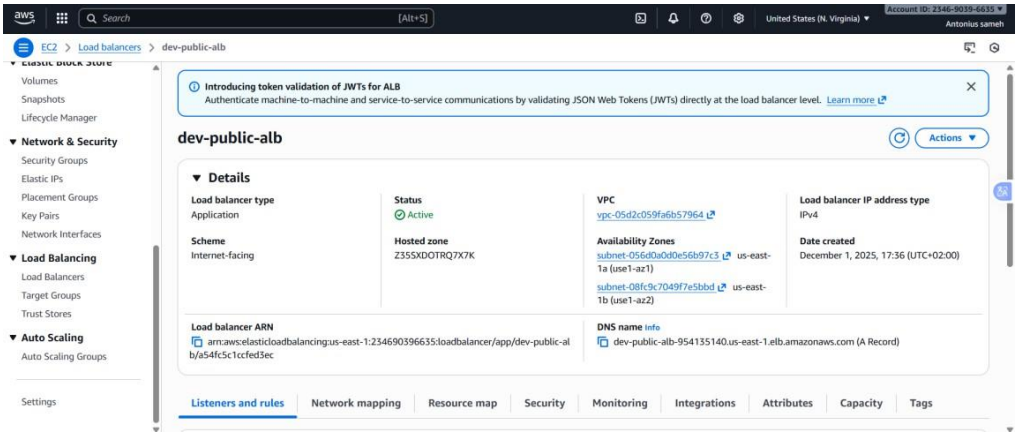

Search [Alt+S]
United States (N. Virginia)
Amazon Elastic Container Service > Task definitions > tts-service-task > Revision 6 > Create revision
Amazon Elastic Container Service
Express Mode New
Clusters
Namespaces
Task definitions
Account settings
Amazon ECR
Repositories
AWS Batch
Documentation
Discover products
Subscriptions
Container - 1 Info
Essential container
Remove
Container details
Specify a name, container image, and whether the container should be marked as essential. Each task definition must have at least one essential container.
Name
tts-service-container
Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.
Essential container
Yes
Image URI
234690396635.dkr.ecr.us-east-1.amazonaws.com/tts-service-repo@sha256:c472c3cf1167800850b9d833107cf801
Browse ECR images
Up to 255 letters (uppercase and lowercase), numbers, hyphens, underscores, colons, periods, forward slashes, and number signs are allowed.
Private registry Info
Store credentials in Secrets Manager, and then use the credentials to reference images in private registries.
Private registry authentication
Port mappings Info
Add port mappings to allow the container to access ports on the host to send or receive traffic. For port name, a default will be assigned if left blank.
Container port
5000
Protocol
TCP
Port name
tts-service
App protocol
HTTP
Remove
Add port mapping
Read only root file system Info



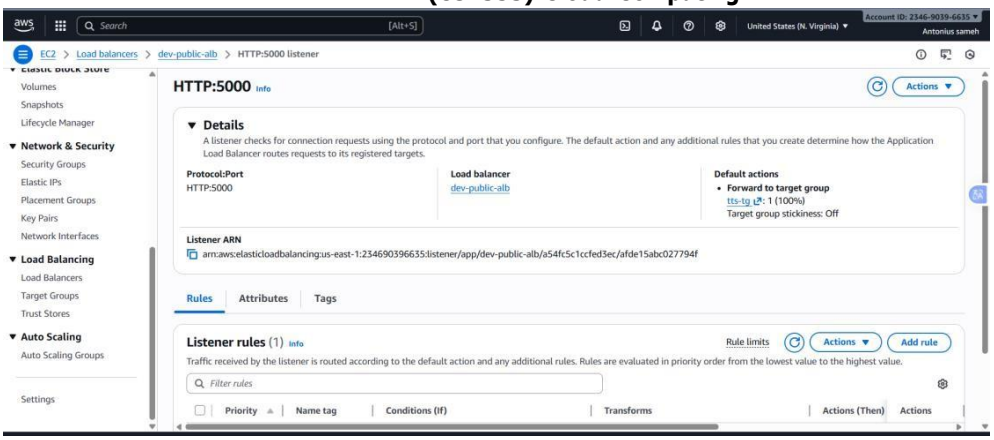
Configure Load Balancer

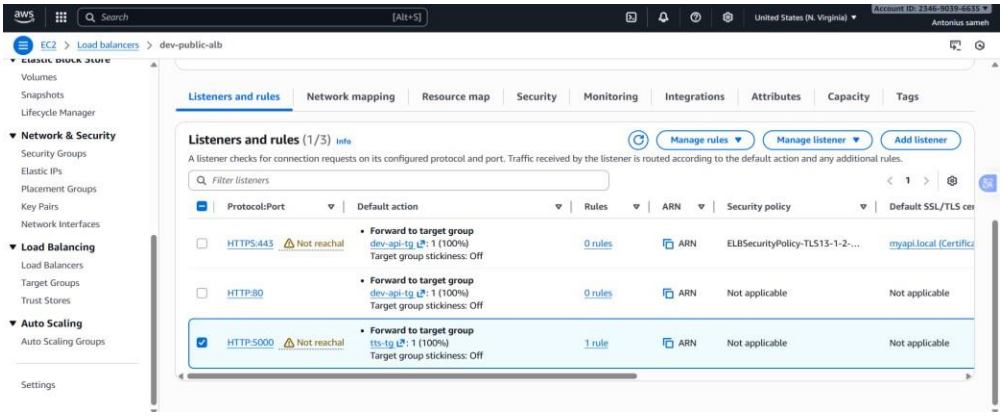
- Listener port: 5000
- Target group:tts-tg
- Type: IP
- Port: 5000
- Health check: /health
- ECS tasks automatically register as targets.

Load Balancer



Listener





Kafka Integration for TTS Microservice

- Create Kafka Cluster (Amazon MSK) Steps:
- Open Amazon MSK in AWS Console
- Click Create Cluster Select:
- Cluster Type: Provisioned
- Number of Brokers: 3
- Instance Type: kafka.m5.large
- Storage: 100–200 GB
- Create or choose a VPC, subnets, and security groups
- Wait for cluster status → "Active"

Create Kafka Topics

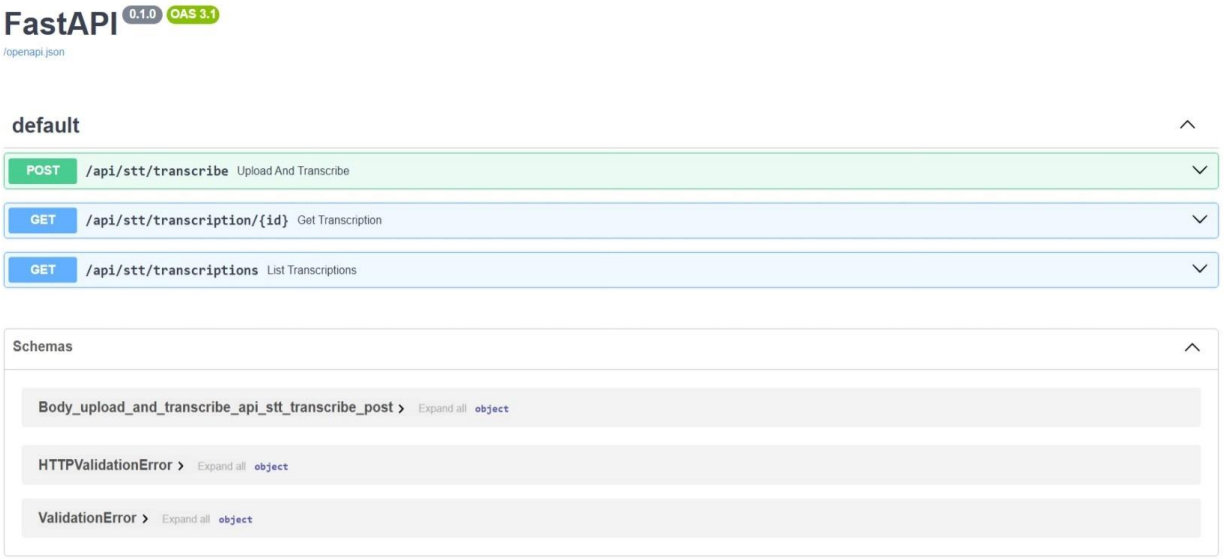
You need two topics:

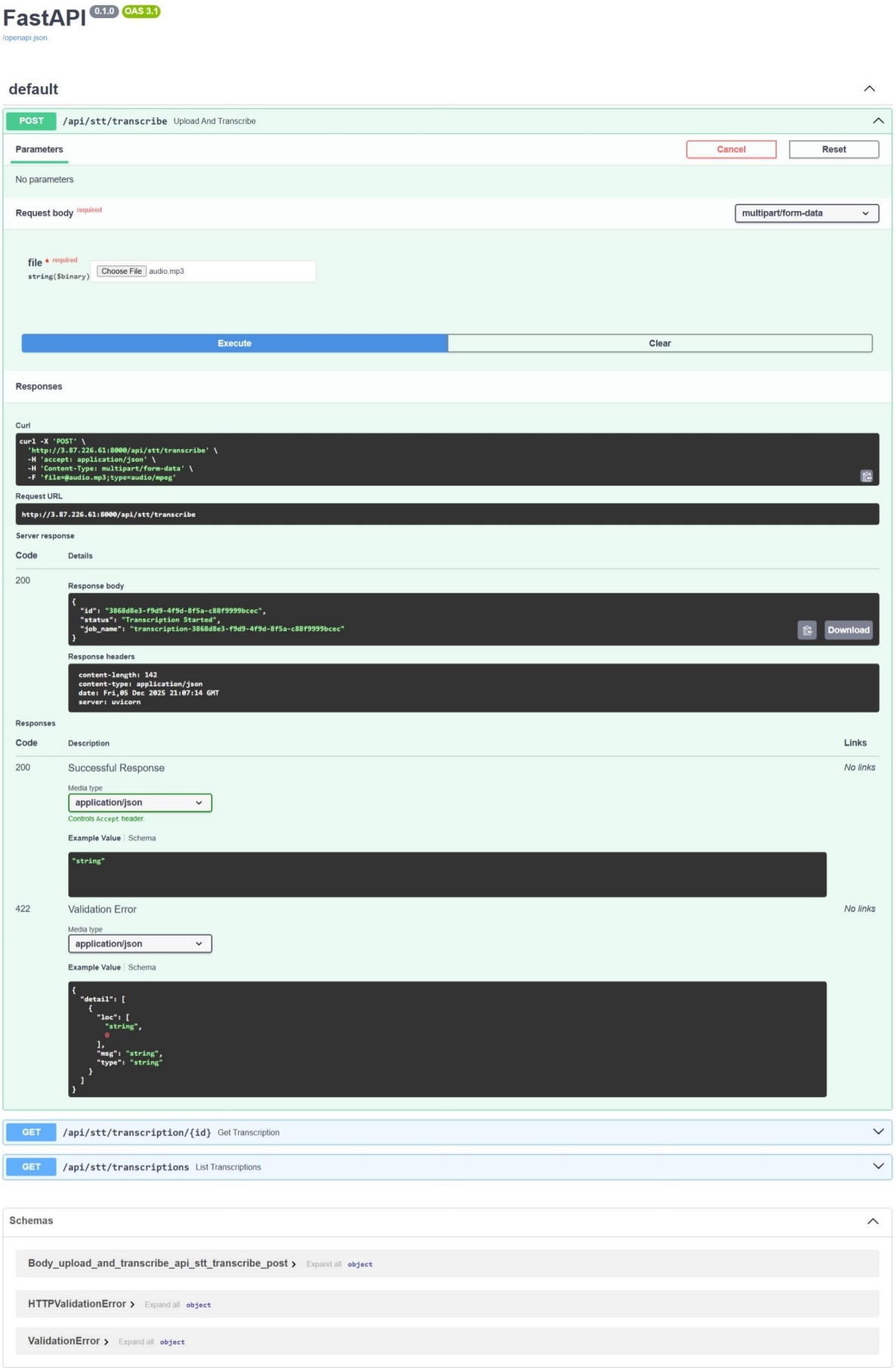
Consumer (TTS) :**audio.generation.requested** to Receive text generation jobs.

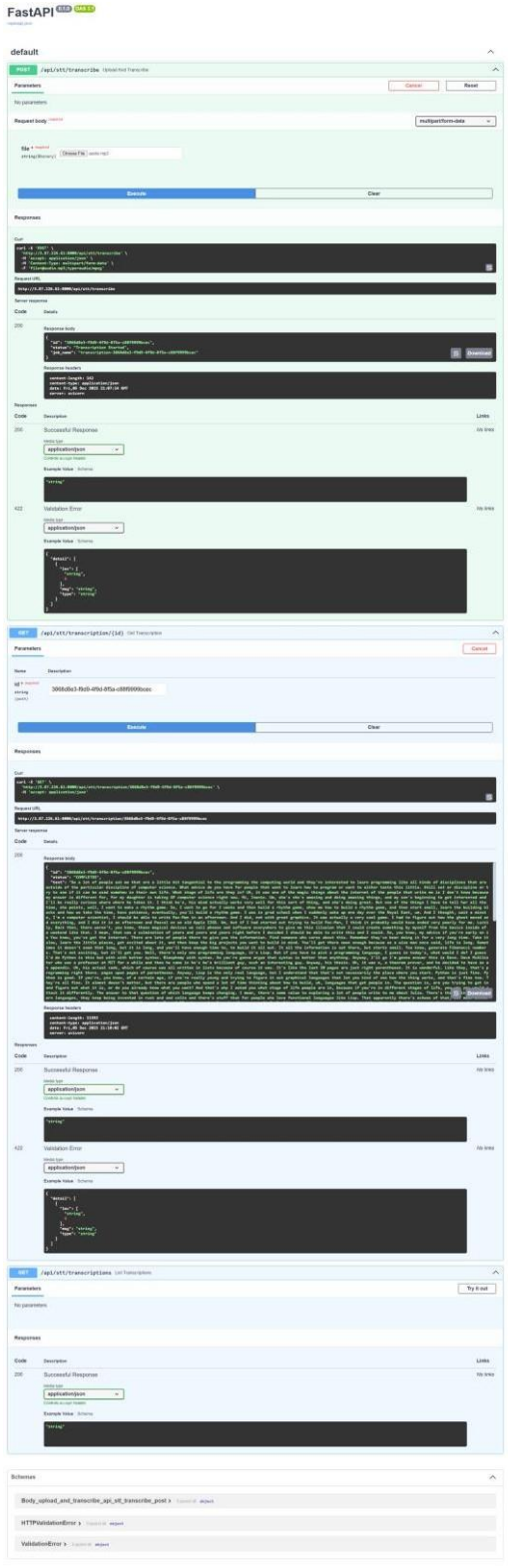
Producer (TTS): **audio.generation.completed** to Send event after audio

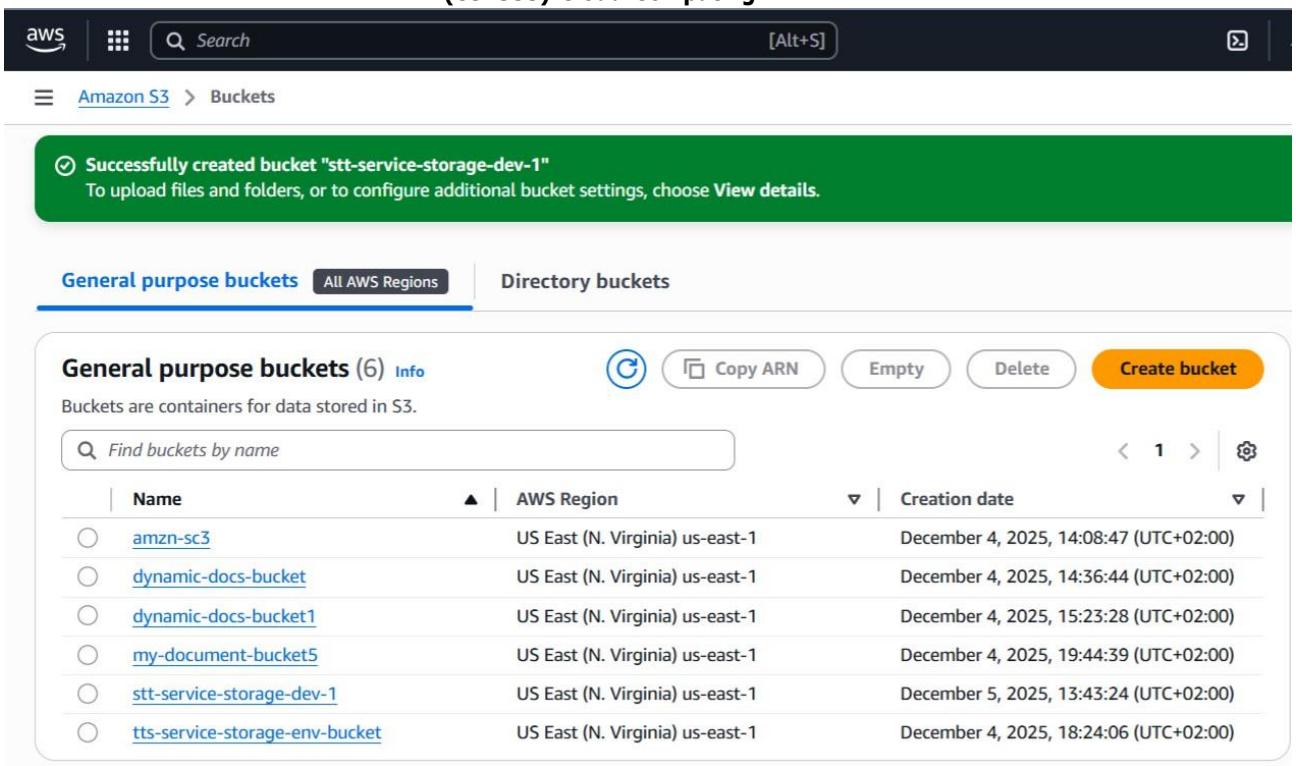
is generated

5.2 Speech-to-Text (STT) Service




FastAPI 0.1.0 OAS 3.1
/openapi.json
default
POST /api/stt/transcribe Upload And Transcribe
Parameters
Cancel
Reset
No parameters
Request body required
multipart/form-data
file * required
string($binary)
Choose File audio.mp3
Execute
Clear
Responses
Curl
curl -X 'POST' \
'http://3.87.226.61:8000/api/stt/transcribe' \
-H 'accept: application/json' \
-H 'Content-Type: multipart/form-data' \
-F 'file=@audio.mp3;type=audio/mpeg'
Request URL
http://3.87.226.61:8000/api/stt/transcribe
Server response
Code
Details
200
Response body
{
"id": "3868d8e3-f9d9-4f9d-8f5a-c88f9999bcec",
"status": "Transcription Started",
"job_name": "transcription-3868d8e3-f9d9-4f9d-8f5a-c88f9999bcec"
}
Download
Response headers
content-length: 142
content-type: application/json
date: Fri,05 Dec 2025 21:07:14 GMT
server: uvicorn
Responses
Code
Description
Links
200
Successful Response
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
"string"
422
Validation Error
No links
Media type
application/json
Example Value | Schema
{
"detail": [
{
"loc": [
"string",
0
],
"msg": "string",
"type": "string"
}
]
}
GET /api/stt/transcription/{id} Get Transcription
GET /api/stt/transcriptions List Transcriptions
Schemas
Body_upload_and_transcribe_api_stt_transcribe_post Expand all object
HTTPValidationError Expand all object
ValidationError Expand all object


FastAPI
default
/api/stt/transcribe
Parameters
Cancel
Reset
No parameters
Request body
multipart/form-data
file *
Choose File
Execute
Clear
Responses
Curl
Request URL
Server response
Code
Details
Response body
Download
Response headers
Code
Description
Links
Successful Response
application/json
Example Value
Schema
"string"
Validation Error
HTTPValidationError
{
"detail": [
{
"loc": [
"string",
0
],
"msg": "string",
"type": "string"
}
]
}
/api/stt/transcription/{id}
Get Transcription
Parameters
Cancel
3065d8e3-fbd3-4fbd-8f5a-c89f9999bcec
Execute
Clear
Responses
Curl
Request URL
Server response
Code
Details
Response body
Download
Response headers
Code
Description
Links
Successful Response
application/json
Example Value
Schema
"string"
Validation Error
HTTPValidationError
{
"detail": [
{
"loc": [
"string",
0
],
"msg": "string",
"type": "string"
}
]
}
/api/stt/transcriptions
Get Transcriptions
Parameters
Try it out
No parameters
Responses
Code
Description
Links
Successful Response
application/json
Example Value
Schema
"string"
Schemas
Body_upload_and_transcribe_api_stt_transcribe_post
HTTPValidationError
ValidationError



5.3 Chat Completion Service

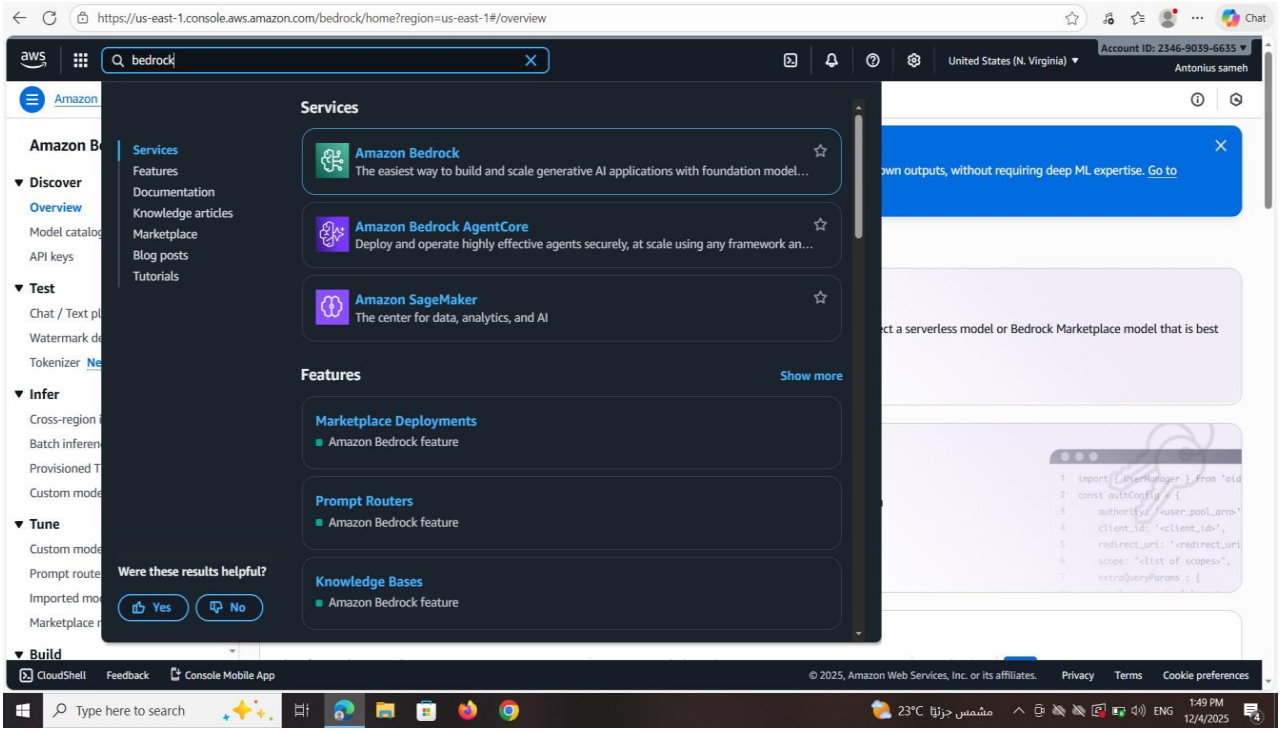
5.4 Document Reader Service

Bedrock Service

1.Accessing the Amazon Bedrock Service

The first step is to locate and navigate to the Amazon Bedrock service within your AWS Management Console.

- **Action:** In the AWS search bar, type bedrock.
- **Result :** The top result will be **Amazon Bedrock**, described as "The easiest way to build and scale generative AI applications with foundation models..."
- **Navigation:** Click on the **Amazon Bedrock** service link to open its main console page.

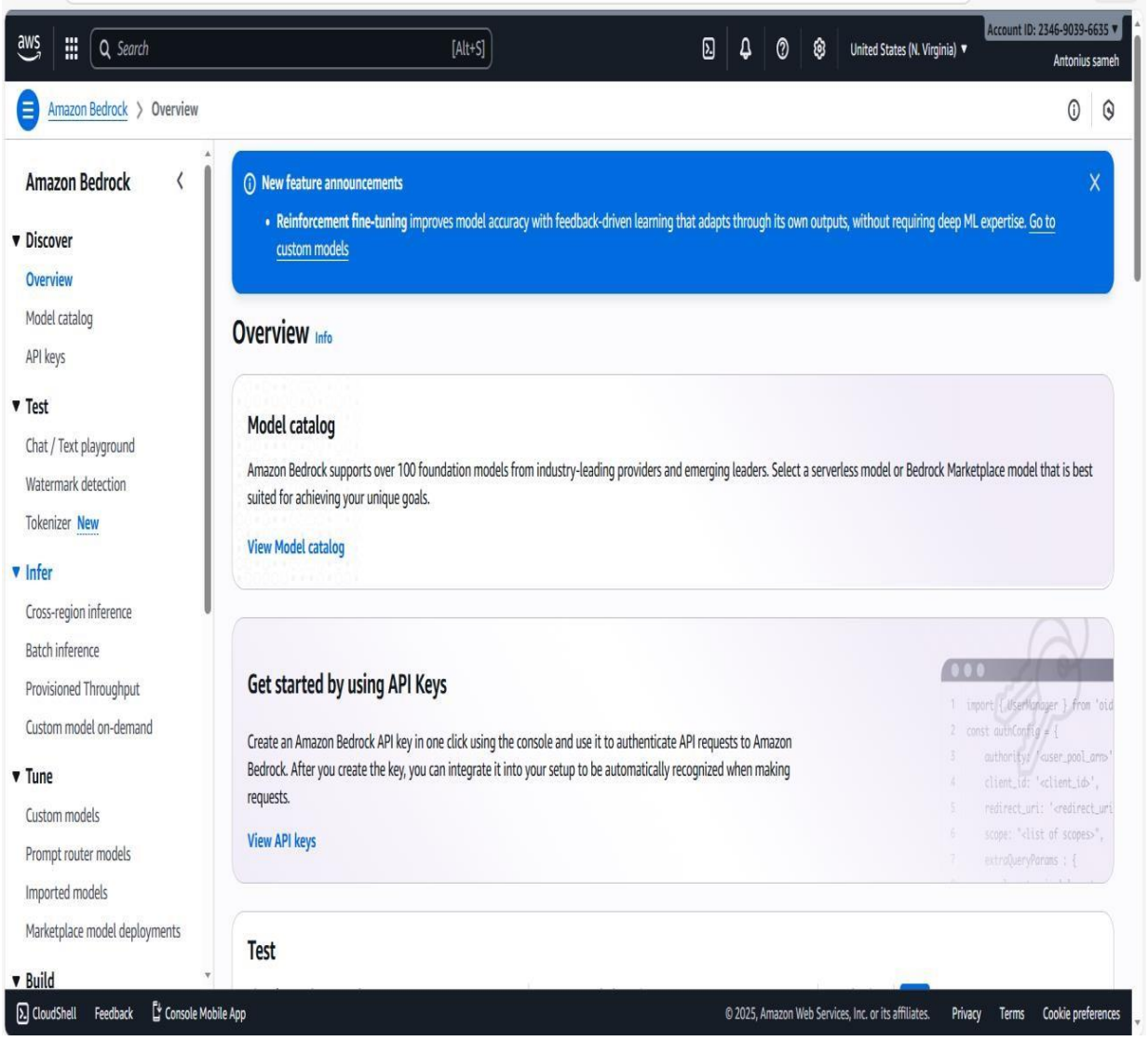


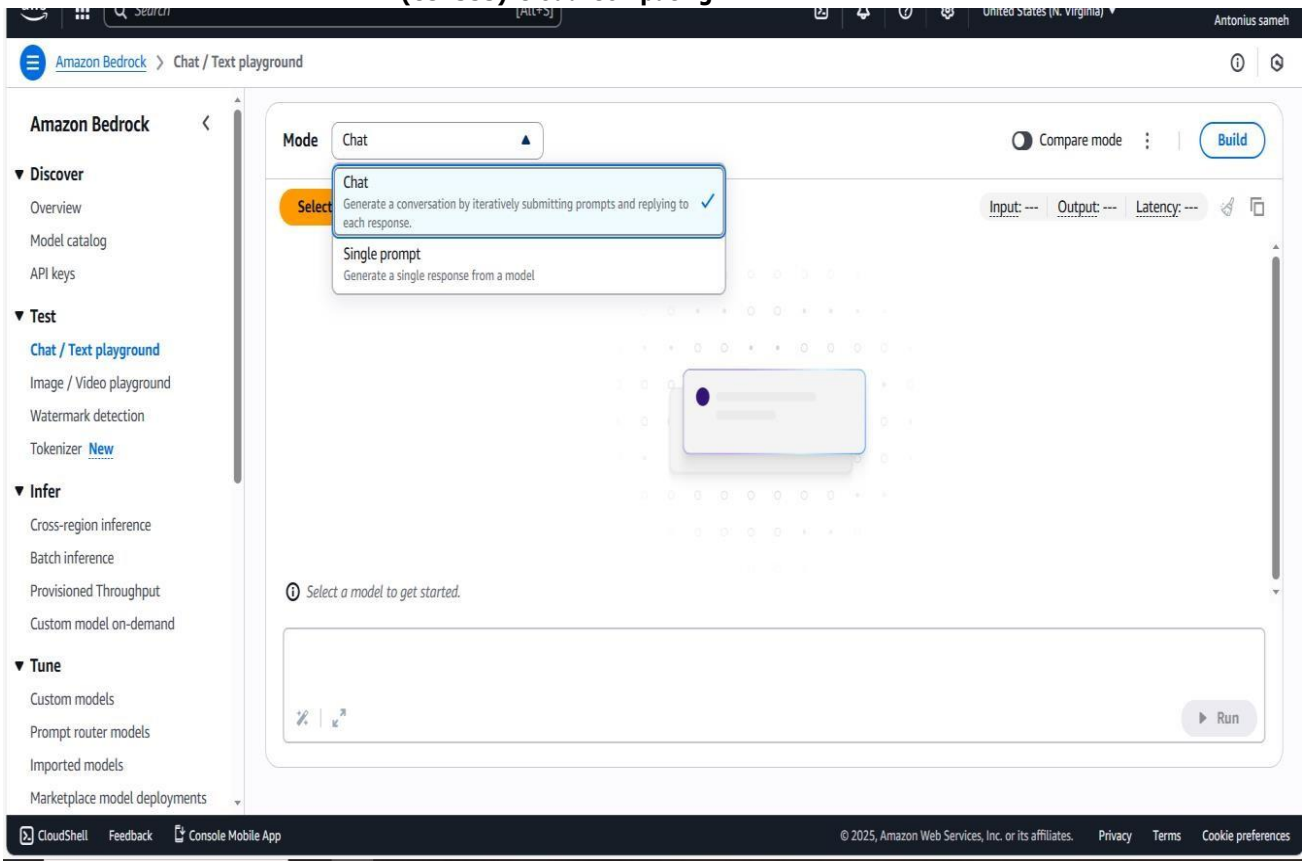
- **Overview and Initial Setup**
  - Once you are on the Amazon Bedrock page, you will see an overview of the service and necessary setup steps.
- The Overview page provides information on **New feature announcements** (like reinforcement fine-tuning), the **Model catalog** (stating Bedrock supports over 100 foundation models), and how to **Get started by using API Keys**.
- **Next Step:** To begin testing models, navigate to the **Test** section in the left-hand menu.
  - **3. Navigating to the Text Playground**
  - The playground is where you can interact directly with the foundation models.
- **Navigation:** In the left-hand navigation pane, under the **Test** section, click on **Chat / Text playground**.
  - **4. Configuring the Text Playground Mode**
  - The Text Playground allows you to choose how you want to interact with the models.
- **Mode Selection :** You have two main modes:
  - **Chat:** Generate a conversation by iteratively submitting prompts and replying to each response. (This is selected in the image).
- **Single prompt:** Generate a single response from a model.

- **Recommended Action:** For an interactive conversation, select the **Chat** mode.

- **5. Selecting a Model to Start Inferencing**

- After setting the mode, the next crucial step is selecting which Foundation Model (FM) you want to use.
  - **Current State :** The playground is ready, but you need to select a model. The prompt reads: **Select a model to get started.**
  - **Action:** Click the **Select model** button at the top of the playground. A list of available models (e.g., from Anthropic, AI21 Labs, Cohere, Amazon) will appear, provided you have requested and been granted **Model access** in the settings.
  - **Final Step:** Once a model is selected, you can type your prompt into the text box at the bottom and click **Run** to generate a response.





Selecting the Foundation Model (FM)

After clicking "Select model," a dialog box appears, allowing you to choose the specific Foundation Model you want to use for inference.

- **Model Selection Dialog :** The selection process is broken down into three columns:
  - **Categories (Model Providers):** Here you choose the company that created the model (e.g., AI21 Labs, Amazon, Anthropic, Meta).
  - **Models:** After selecting a provider (e.g., **Meta** is highlighted), you choose the specific model version (e.g., **Llama 3.1 70B Instruct v1** is selected in the image). The token limits are also displayed here (e.g., **128K tokens**).
  - **Inference:** This section typically confirms the chosen inference profile or region (e.g., **US Meta Llama 3.1 70B Instruct**).
- **Action:** Select your desired model and click the **Apply** button to load it into the playground.

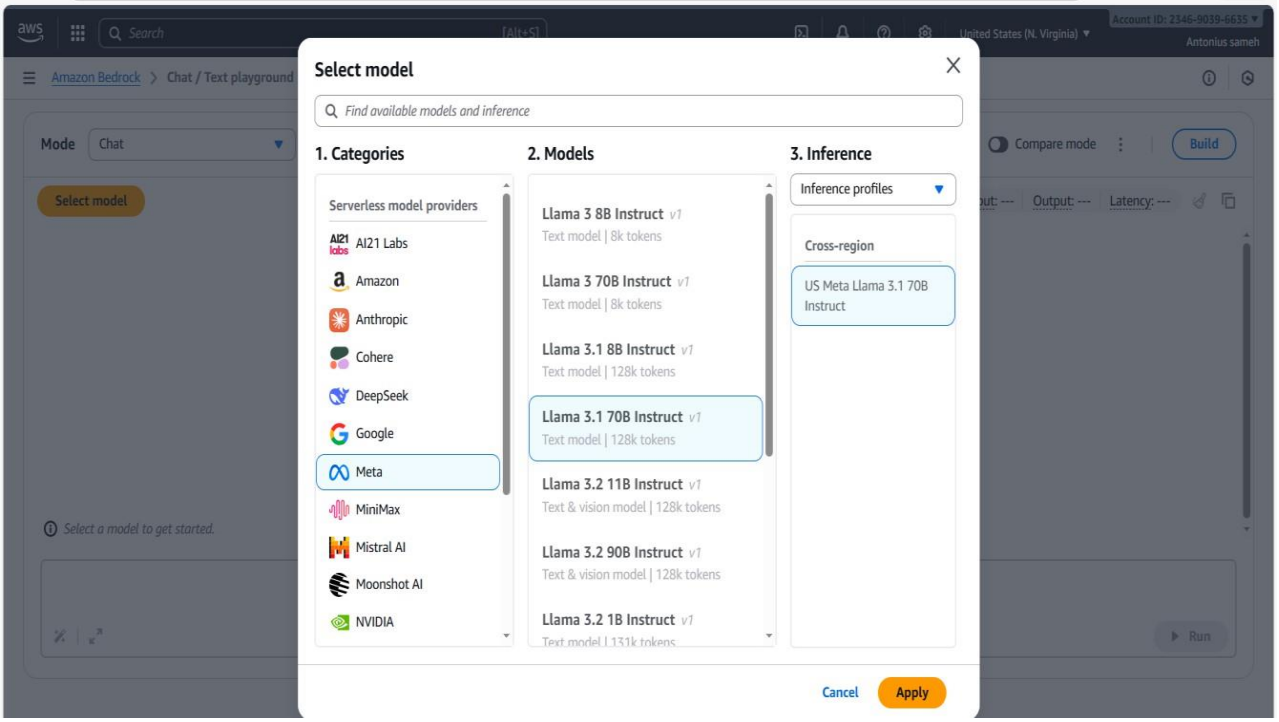
7. Interacting with the Selected Model

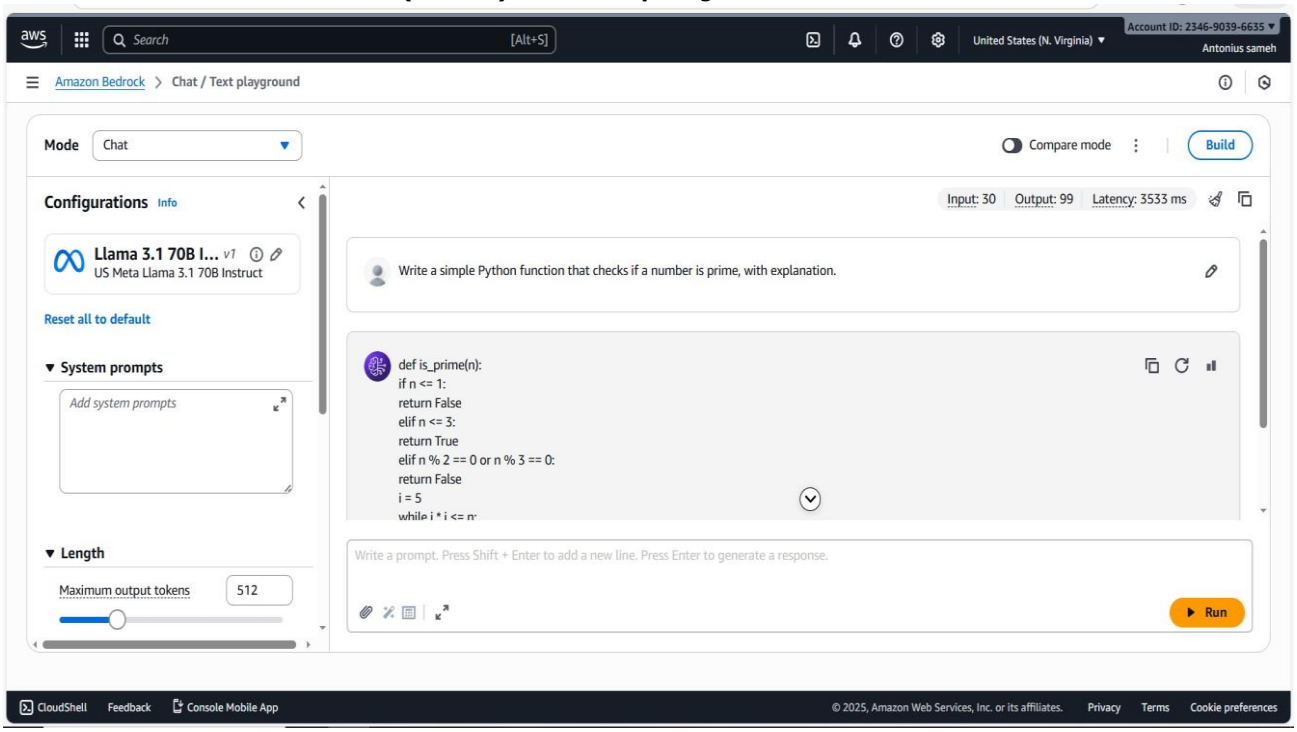
Once the model is loaded, the playground is ready for you to send prompts and receive generated output.

- **Loaded Model :** The selected model, **Llama 3.1 70B Instruct v1**, is

now displayed under **Configurations**.

- **Model Configurations:** On the left, you can adjust inference parameters like:
  - **System prompts:** To provide context or instructions to guide the model's behavior.
  - **Length (Maximum output tokens):** To set a limit on the length of the model's response (e.g., **512** tokens).
- **Prompting and Generation:**
  - **Input:** The user has provided a prompt: "Write a simple Python function that checks if a number is prime, with explanation."
  - **Output:** The model has generated the requested Python code (def is_prime(n):...) along with the explanation (implied by the presence of the response box).
  - **Metrics:** The top right shows performance metrics: **Input** (30 tokens), **Output** (99 tokens), and **Latency** (3533 ms).
- **Action:** To continue the conversation or run a new prompt, type your next text into the box and click **Run**.





Building the Docker Image

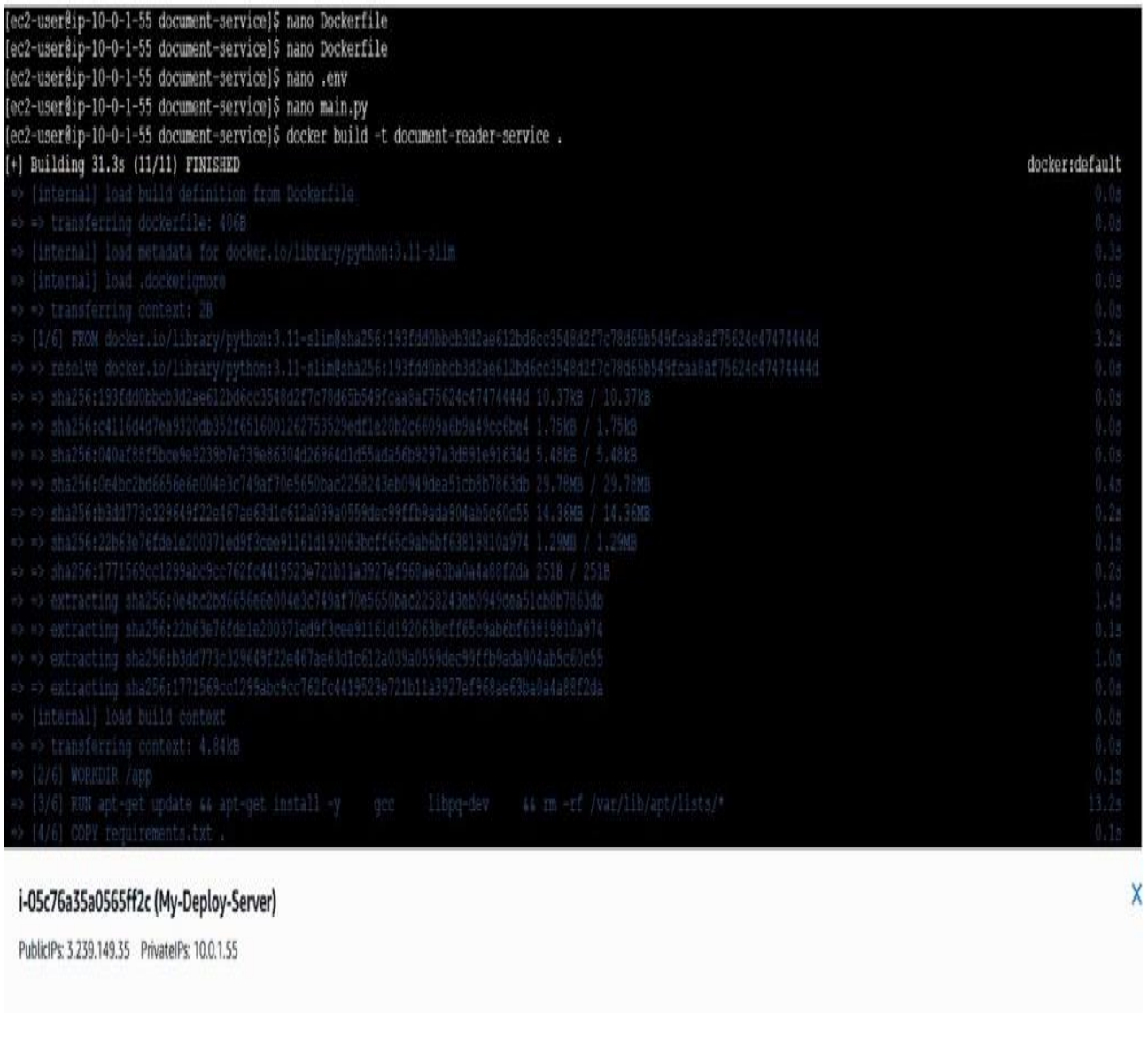
The service is prepared and built on an EC2 instance (i-05c76a35a0565ff2c).

- **Setup**
  - The necessary project files, including a requirements.txt file listing Python dependencies like fastapi, uvicorn, boto3, and PyPDF2, are created.
  - A Dockerfile is prepared to define the container environment.

Building the Image :

- The command docker build -t document-reader-service . is executed.
- This command builds the Docker image locally, tagging it as document-reader-service.
- The build steps include loading the Dockerfile, pulling the base Python image (python:3.11-slim), installing dependencies (e.g., gcc, libpq-dev via apt-get), and copying requirements.
- **Result:** The build process completes successfully .

- 
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023
Last login: Thu Dec 4 20:44:05 2025 from 18.206.107.27
[ec2-user@ip-10-0-1-55 ~]$ cd document-service
[ec2-user@ip-10-0-1-55 document-service]$ nano requirements.txt
[ec2-user@ip-10-0-1-55 document-service]$ nano requirements.txt
[ec2-user@ip-10-0-1-55 document-service]$ cat requirements.txt
cat: requirements.txt: No such file or directory
[ec2-user@ip-10-0-1-55 document-service]$ nano requirements.txt
[ec2-user@ip-10-0-1-55 document-service]$ cat requirements.txt
fastapi==0.109.0
uvicorn==0.27.0
python-multipart==0.0.6
boto3==1.34.14
psycopg2-binary==2.9.9
kafka-python==2.0.2
PyPDF2==3.0.1
python-docx==1.1.0
openai==1.10.0
python-dotenv==1.0.0
[ec2-user@ip-10-0-1-55 document-service]$ nano Dockerfile


[ec2-user@ip-10-0-1-55 document-service]$ nano Dockerfile
[ec2-user@ip-10-0-1-55 document-service]$ nano Dockerfile
[ec2-user@ip-10-0-1-55 document-service]$ nano .env
[ec2-user@ip-10-0-1-55 document-service]$ nano main.py
[ec2-user@ip-10-0-1-55 document-service]$ docker build -t document-reader-service .
[+] Building 31.3s (11/11) FINISHED
docker:default
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 406B
=> [internal] load metadata for docker.io/library/python:3.11-slim
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/6] FROM docker.io/library/python:3.11-slim@sha256:193fdd0bbcb3d2ae612bd6cc3548d2f7c78d65b549fcaa8af75624c474744444d
=> => resolve docker.io/library/python:3.11-slim@sha256:193fdd0bbcb3d2ae612bd6cc3548d2f7c78d65b549fcaa8af75624c474744444d
=> => sha256:193fdd0bbcb3d2ae612bd6cc3548d2f7c78d65b549fcaa8af75624c474744444d 10.37kB / 10.37kB
=> => sha256:c4116d4d7ea9320db352f6516001262753529edf1e20b2c6609a0b9a49cc6be4 1.75kB / 1.75kB
=> => sha256:040af88f5bce9e9239b7e739e86304d26964d1d55ada56b9297a3d891e91634d 5.48kB / 5.48kB
=> => sha256:0e4bc2bd6656e6e004e3c749af70e5650bac2258243eb0949dea51cb8b7863db 29.78MB / 29.78MB
=> => sha256:b3dd773c329649f22e467ae63d1c612a039a0559dec99ffb9ada904ab5c60e55 14.36MB / 14.36MB
=> => sha256:22b63e76fde1e200371ed9f3cee91161d192063bcff65c9ab6bf63819810a974 1.29MB / 1.29MB
=> => sha256:1771569cc1299abc9cc762fc4419523e721b11a3927ef968ae63ba0a4a88f2da 251B / 251B
=> => extracting sha256:0e4bc2bd6656e6e004e3c749af70e5650bac2258243eb0949dea51cb8b7863db
=> => extracting sha256:22b63e76fde1e200371ed9f3cee91161d192063bcff65c9ab6bf63819810a974
=> => extracting sha256:b3dd773c329649f22e467ae63d1c612a039a0559dec99ffb9ada904ab5c60e55
=> => extracting sha256:1771569cc1299abc9cc762fc4419523e721b11a3927ef968ae63ba0a4a88f2da
=> [internal] load build context
=> => transferring context: 4.84kB
=> [2/6] WORKDIR /app
=> [3/6] RUN apt-get update && apt-get install -y gcc libpq-dev && rm -rf /var/lib/apt/lists/*
=> [4/6] COPY requirements.txt .

i-05c76a35a0565ff2c (My-Deploy-Server)
PublicIPs: 3.239.149.35 PrivateIPs: 10.0.1.55

Running the Container :

- The command docker run --name document-service -p 8000:8000 --env-file .env document-reader-service is used to:
  - Run the image tagged document-reader-service.
  - Map port 8000 on the container to port 8000 on the EC2 host.
  - Set the container name as document-service.
  - Load environment variables from a local .env file.

- **Testing the API :**
  - The command curl http://localhost:8000/openapi.json is executed to hit

the service's API documentation endpoint.

- **Result:** The command successfully returns the full OpenAPI

JSON specification, confirming the service is running and accessible.

. Creating the ECR Repository

The ECR repository is a secure location to store the Docker image on AWS.

- **Initial Push Attempt & Failure :**
  - An initial attempt to run docker push fails with the error "The repository with name 'document-reader-service' does not exist in the registry..."
  - Creating the Repository :
- The command aws ecr create-repository --repository-name document-reader-service --region us-east-1 is executed.
  - **Result:** The repository is successfully created, returning a JSON object confirming the **repositoryUri** (234690396635.dkr.ecr.us-east-1.amazonaws.com/document-reader-service).

```
[ec2-user@ip-10-0-1-55 document-service]$ docker push 234690396635.dkr.ecr.us-east-1.amazonaws.com/document-reader-service:latest
The push refers to repository [234690396635.dkr.ecr.us-east-1.amazonaws.com/document-reader-service]
b19d4dca9597: Preparing
f268ceb4f6bf: Preparing
3734a5ac2887: Preparing
5377a646f4a0: Preparing
8464b0b77e61: Preparing
720ee7aae3ad: Waiting
655ff69eb9c8: Waiting
5c988eaa8862: Waiting
70a290c5e58b: Waiting
name unknown: The repository with name 'document-reader-service' does not exist in the registry with id '234690396635'
[ec2-user@ip-10-0-1-55 document-service]$ aws ecr create-repository --repository-name document-reader-service --region us-east-1
{
    "repository": {
        "repositoryArn": "arn:aws:ecr:us-east-1:234690396635:repository/document-reader-service",
        "registryId": "234690396635",
        "repositoryName": "document-reader-service",
        "repositoryUri": "234690396635.dkr.ecr.us-east-1.amazonaws.com/document-reader-service",
        "createdAt": "2025-12-04T21:30:07.733000+00:00",
        "imageTagMutability": "MUTABLE",
        "imageScanningConfiguration": {
            "scanOnPush": false
        },
        "encryptionConfiguration": {
            "encryptionType": "AES256"
        }
    }
}
```

i-05c76a35a0565ff2c (My-Deploy-Server)

PublicIPs: 3.239.149.35 PrivateIPs: 10.0.1.55

X

```
[ec2-user@ip-10-0-1-55 document-service]$ docker run -d --name document-service -p 8000:8000 --env-file .env document-reader-service
dc570963f7dd8c9584eab27315901ee43bea60c92f18dceb6ea05add96b3026b
[ec2-user@ip-10-0-1-55 document-service]$ curl http://localhost:8000/openapi.json
{"openapi":"3.1.0","info":{"title":"Document Reader Service","version":"0.1.0"},"paths":{"/api/documents/upload":{"post":{"summary":"Upload Document","operationId":"up
load_document_api_documents_upload_post","requestBody":{"content":{"multipart/form-data":{"schema":{"$ref":"#/components/schemas/Body_upload_document_api_documents_upl
oad_post"}}},"required":true},"responses":{"200":{"description":"Successful Response","content":{"application/json":{"schema":{}}}},"422":{"description":"Validation Er
ror","content":{"application/json":{"schema":{"$ref":"#/components/schemas/HTTPValidationError"}}}}}}},"/api/documents/{doc_id}":{"get":{"summary":"Get Document Detail
s","operationId":"get_document_details_api_documents__doc_id__get","parameters":[{"name":"doc_id","in":"path","required":true,"schema":{"type":"string","title":"Doc Id
"}}],"responses":{"200":{"description":"Successful Response","content":{"application/json":{"schema":{}}}},"422":{"description":"Validation Error","content":{"applicat
ion/json":{"schema":{"$ref":"#/components/schemas/HTTPValidationError"}}}}},"delete":{"summary":"Delete Document","operationId":"delete_document_api_documents__doc_id
__delete","parameters":[{"name":"doc_id","in":"path","required":true,"schema":{"type":"string","title":"Doc Id"}}],"responses":{"200":{"description":"Successful Respon
se","content":{"application/json":{"schema":{}}}},"422":{"description":"Validation Error","content":{"application/json":{"schema":{"$ref":"#/components/schemas/HTTPVal
idationError"}}}}}}},"/api/documents/{doc_id}/notes":{"get":{"summary":"Get Document Notes","operationId":"get_document_notes_api_documents__doc_id__notes_get","parame
ters":[{"name":"doc_id","in":"path","required":true,"schema":{"type":"string","title":"Doc Id"}}],"responses":{"200":{"description":"Successful Response","content":{"a
pplication/json":{"schema":{}}}},"422":{"description":"Validation Error","content":{"application/json":{"schema":{"$ref":"#/components/schemas/HTTPValidationError"}}}}
}}},"/api/documents/{doc_id}/regenerate-notes":{"post":{"summary":"Regenerate Notes","operationId":"regenerate_notes_api_documents__doc_id__regenerate_notes_post","par
ameters":[{"name":"doc_id","in":"path","required":true,"schema":{"type":"string","title":"Doc Id"}}],"responses":{"200":{"description":"Successful Response","content":
{"application/json":{"schema":{}}}},"422":{"description":"Validation Error","content":{"application/json":{"schema":{"$ref":"#/components/schemas/HTTPValidationError"}
}}}}}},"/api/documents":{"get":{"summary":"List Documents","operationId":"list_documents_api_documents_get","responses":{"200":{"description":"Successful Response","co
ntent":{"application/json":{"schema":{}}}}}}},"components":{"schemas":{"Body_upload_document_api_documents_upload_post":{"properties":{"file":{"type":"string","format
":"binary","title":"File"}},"type":"object","required":["file"],"title":"Body_upload_document_api_documents_upload_post"},"HTTPValidationError":{"properties":{"detail"
:{"items":{"$ref":"#/components/schemas/ValidationError"},"type":"array","title":"Detail"}},"type":"object","title":"HTTPValidationError"},"ValidationError":{"properti
es":{"loc":{"items":{"anyOf":[{"type":"string"},{"type":"integer"}]},"type":"array","title":"Location"},"msg":{"type":"string","title":"Message"},"type":{"type":"strin
g","title":"Error Type"}},"type":"object","required":["loc","msg","type"],"title":"ValidationError"}}}}[ec2-user@ip-10-0-1-55 document-service]$
[ec2-user@ip-10-0-1-55 document-service]$ curl http://localhost:8000/openapi.json | grep "regenerate-notes"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  3443  100  3443    0     0  1450k      0 --:--:-- --:--:-- --:--:-- 1681k
```

i-05c76a35a0565ff2c (My-Deploy-Server)

PublicIPs: 3.239.149.35 PrivateIPs: 10.0.1.55

X

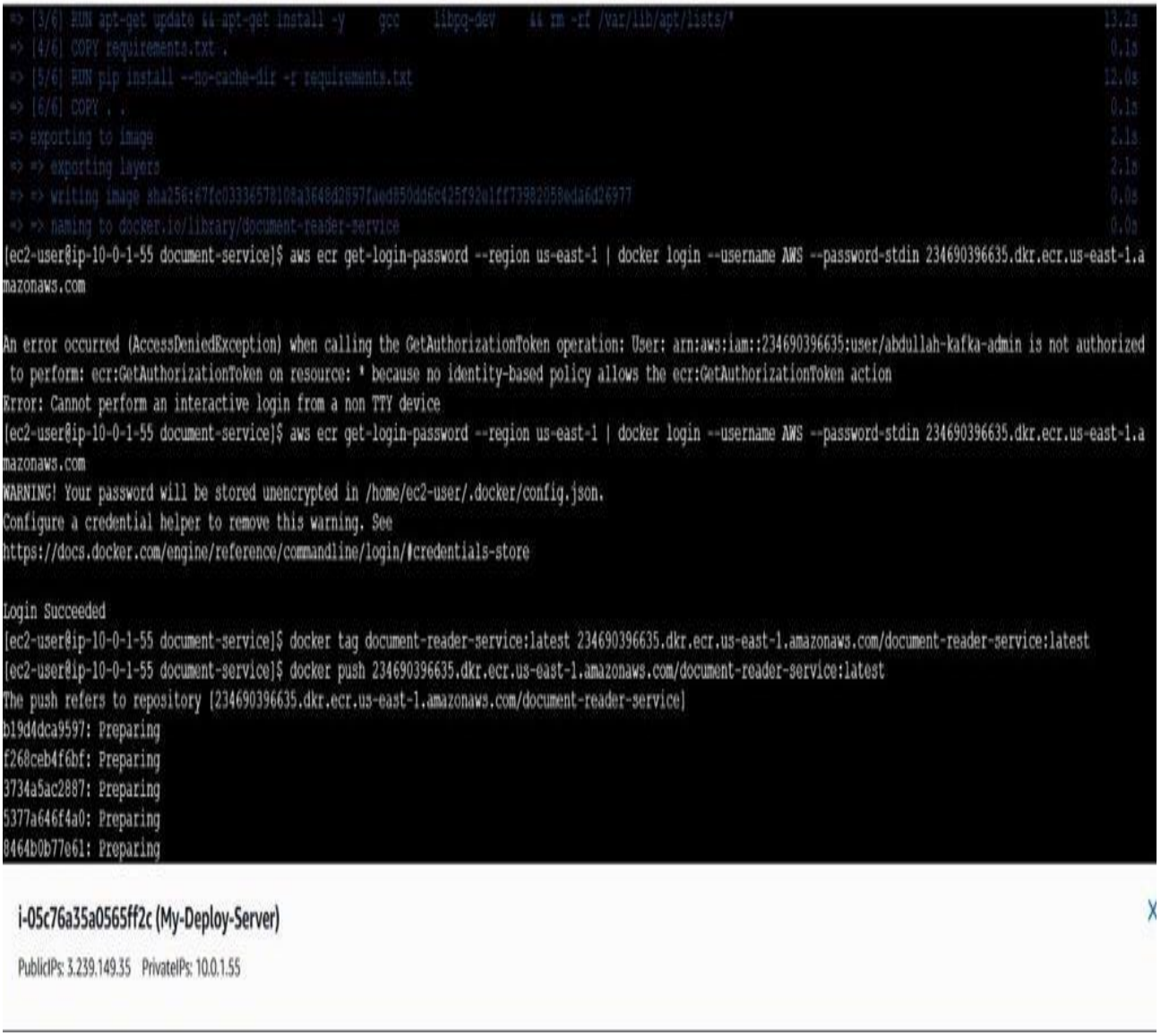
Logging In and Pushing the Image to ECR

The final steps involve authenticating with ECR and uploading the built image.

- Authentication Issues :
  - An initial attempt to authenticate fails with an AccessDeniedException because the IAM user (noura-kafka- admin) lacks the necessary permissions to call GetAuthorizationToken.
- Successful Login :
  - A subsequent successful login is shown after the permission issue is presumably resolved: Login Succeeded.
  - A warning about unencrypted passwords in the Docker configuration is displayed, with a link to configure a credential helper.
- Tagging the Image :
  - The local image is tagged with the ECR repository URI: docker tag document-reader-service:latest 234690396635.dkr.ecr.us- east-1.amazonaws.com/document-reader-service:latest.
- Pushing the Image :
  - The command docker push 234690396635.dkr.ecr.us-

east- 1.amazonaws.com/document-reader-service:latest is executed.

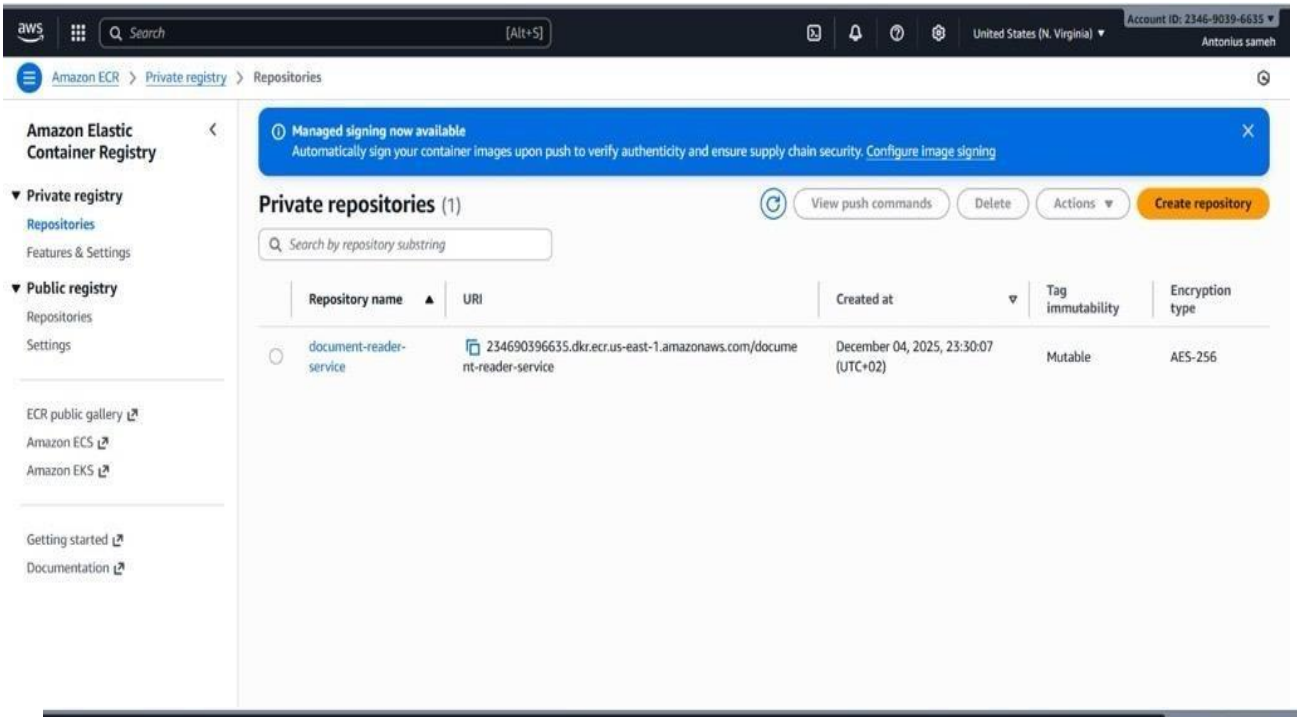
- **Result:** The layers of the image are uploaded successfully (Preparing, Waiting, and implicitly Pushed once all layers are done).

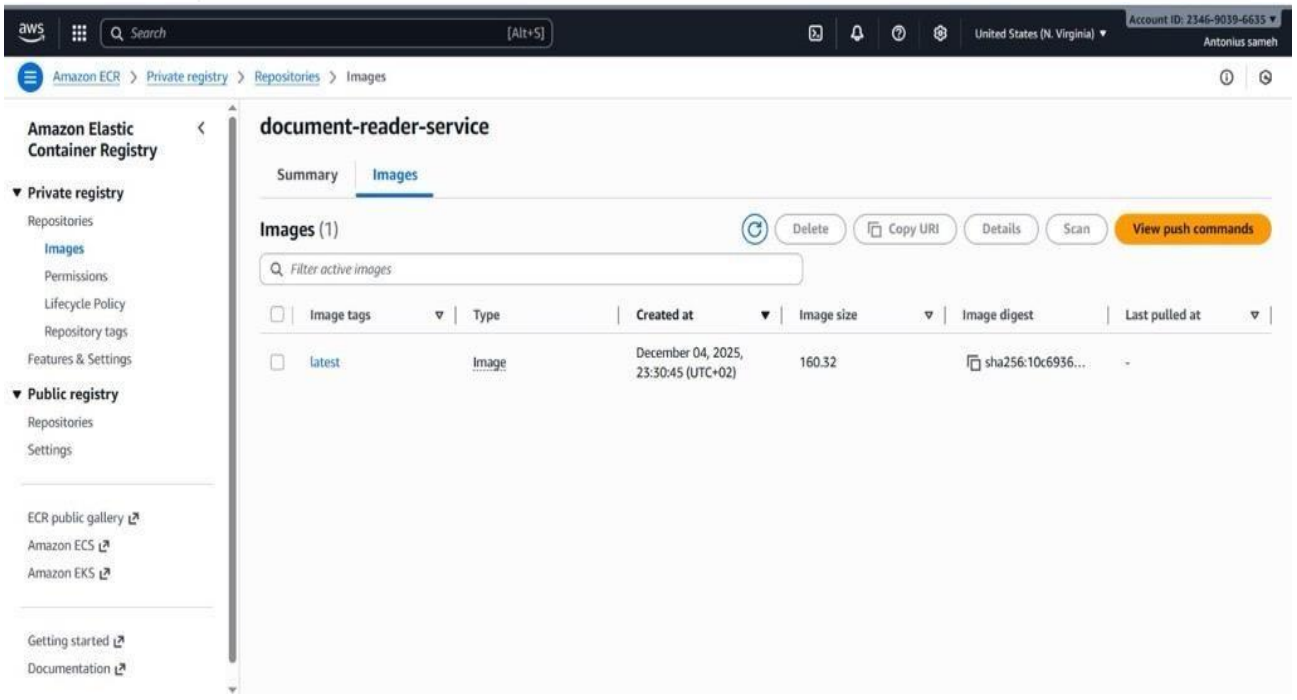


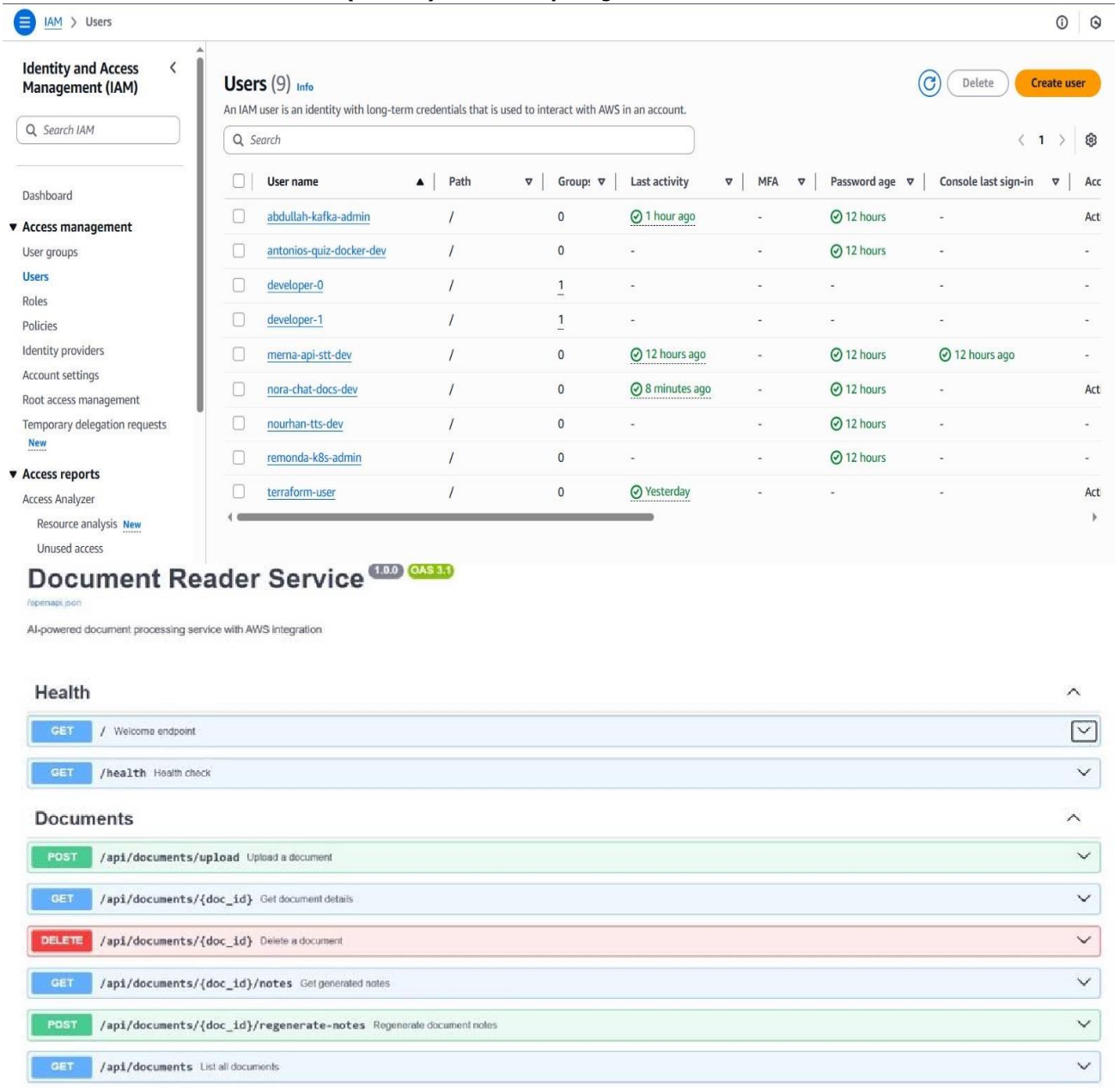
Verifying the Image in the AWS Console

The final step is to confirm the image is present in the ECR console.

- **ECR Console View :**
  - The **Private repositories** page shows the document-reader- service repository and its URI.
- Navigating to the repository details under the **Images** tab confirms the upload.
  - **Verification :** The image with the tag **latest** is listed, showing the creation date (December 04, 2025), size (160.32 MB), and image digest.


aws
Search
[Alt+S]
United States (N. Virginia)
Account ID: 2346-9039-6635
Antonius sameh
Amazon ECR > Private registry > Repositories
Amazon Elastic Container Registry
Private registry
Repositories
Features & Settings
Public registry
Repositories
Settings
ECR public gallery
Amazon ECS
Amazon EKS
Getting started
Documentation
Managed signing now available
Automatically sign your container images upon push to verify authenticity and ensure supply chain security. Configure image signing
Private repositories (1)
View push commands
Delete
Actions
Create repository
Search by repository substring
Repository name
URI
Created at
Tag immutability
Encryption type
document-reader-service
234690396635.dkr.ecr.us-east-1.amazonaws.com/document-reader-service
December 04, 2025, 23:30:07 (UTC+02)
Mutable
AES-256


aws
Search
[Alt+S]
United States (N. Virginia)
Account ID: 2346-9039-6635
Antonius sameh
Amazon ECR > Private registry > Repositories > Images
Amazon Elastic Container Registry
Private registry
Repositories
Images
Permissions
Lifecycle Policy
Repository tags
Features & Settings
Public registry
Repositories
Settings
ECR public gallery
Amazon ECS
Amazon EKS
Getting started
Documentation
document-reader-service
Summary
Images
Images (1)
Delete
Copy URI
Details
Scan
View push commands
Filter active images
Image tags
Type
Created at
Image size
Image digest
Last pulled at
latest
Image
December 04, 2025, 23:30:45 (UTC+02)
160.32
sha256:10c6936...



http://localhost:8000/docs

5.5 Quiz and Exercise Service

The **Quiz and Exercise Service** is a web application designed to generate quizzes and exercises based on input text or predefined content. This service is powered by advanced machine learning models and is typically integrated into educational platforms to assess the understanding of students or users.

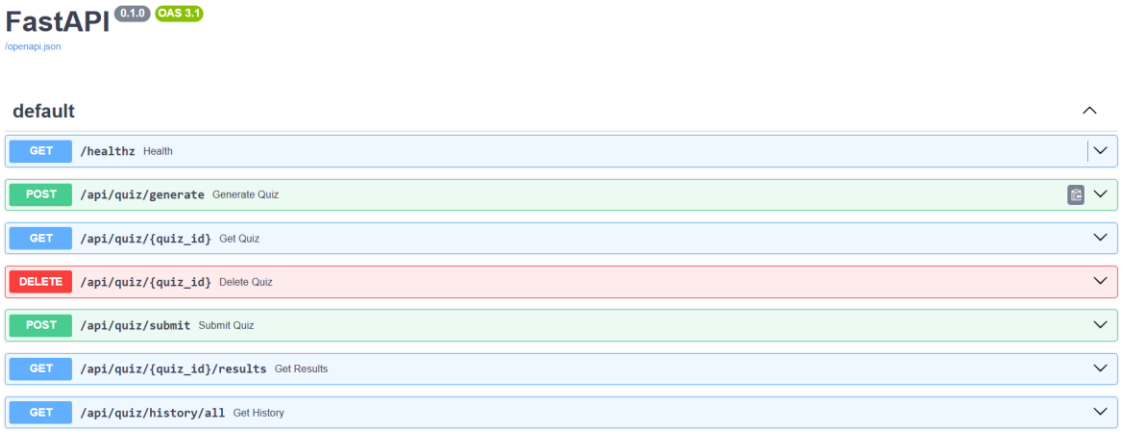
Key Benefits of Docker Implementation:

1. **Isolation**: Each service (e.g., the web application, database, or caching system) can run in its container, ensuring that they do not interfere with each other.
2. **Consistency**: Docker ensures that the application runs in the same environment across different machines, preventing environment-specific bugs.
3. **Scalability**: Docker allows for easy scaling of services. For instance, if the quiz service experiences high traffic, more containers can be spun up to handle the load.
4. **Efficiency**: Docker containers are lightweight, so they consume fewer resources compared to traditional virtual machines.

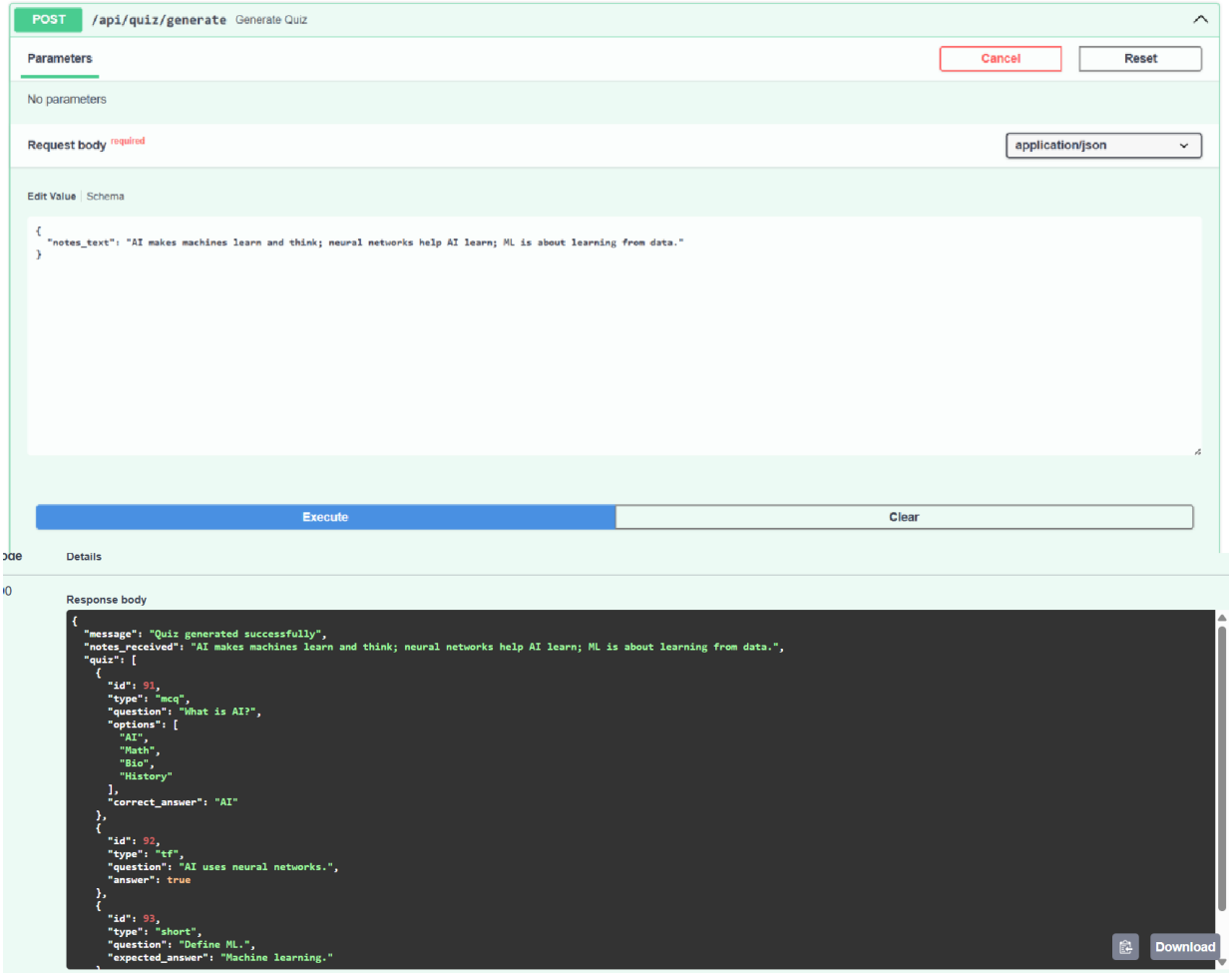
Docker Compose for Multi-Container Setup:

To manage the various services such as the web application, database, and message brokers, **Docker Compose** is used. This tool helps define and run multi-container Docker applications.

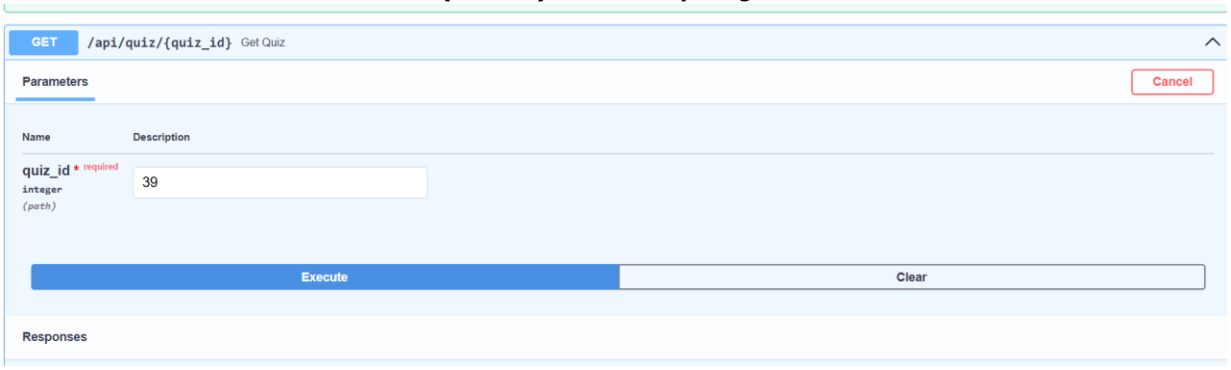
- **docker-compose.yml** defines all the services that need to run in containers.
- **FastAPI Service**: The main service running the quiz generation logic.
- **PostgreSQL Database**: Stores quiz data, questions, and user responses.
- **Kafka**: Used for message streaming (e.g., real-time quiz generation).

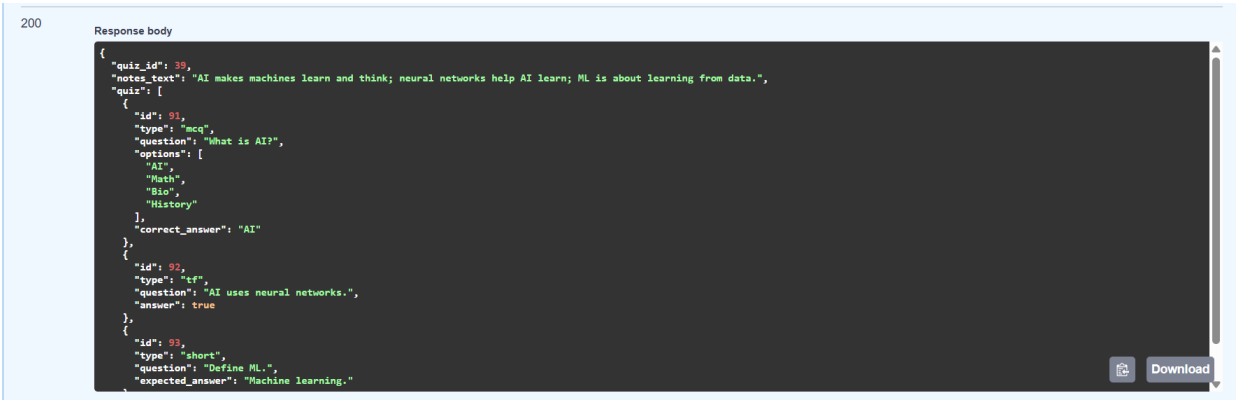


Generate Quiz

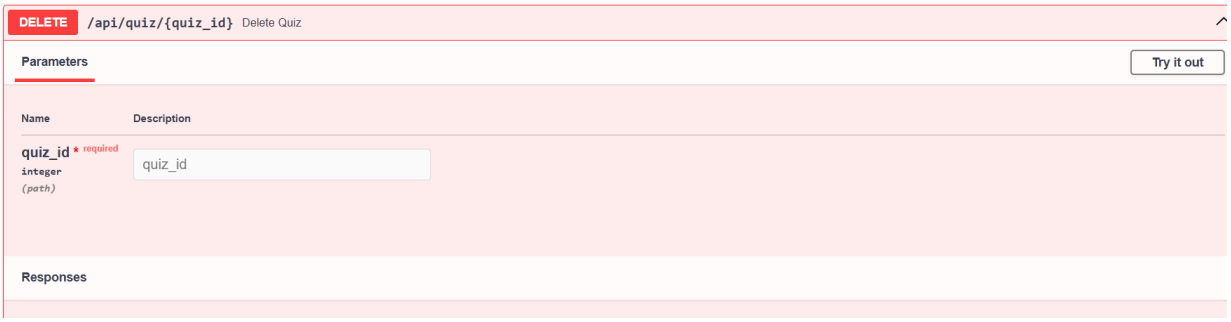


Get Quiz

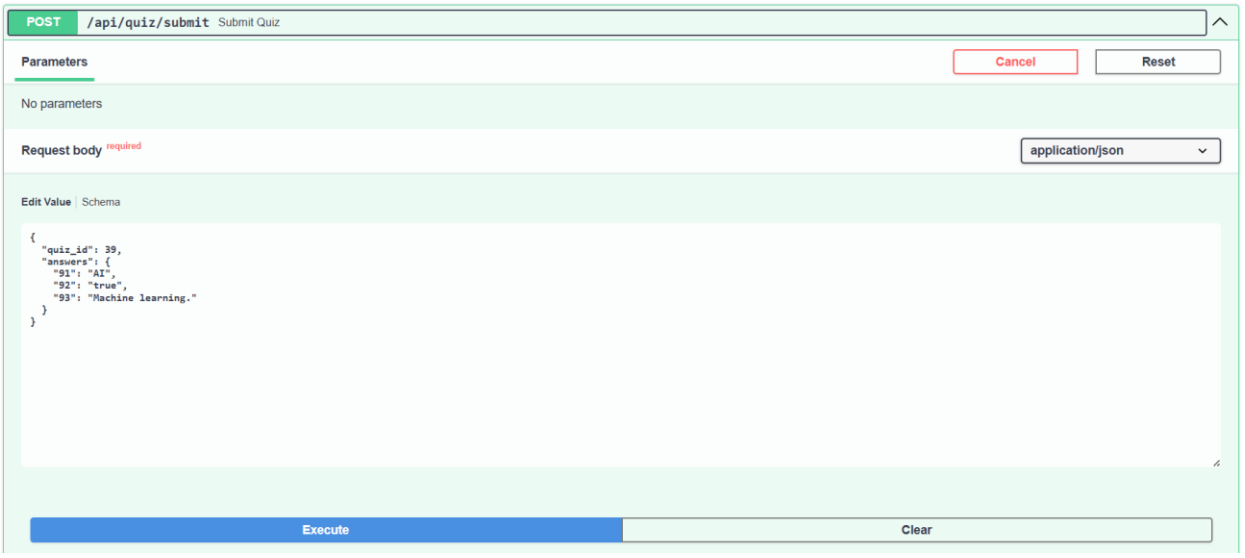


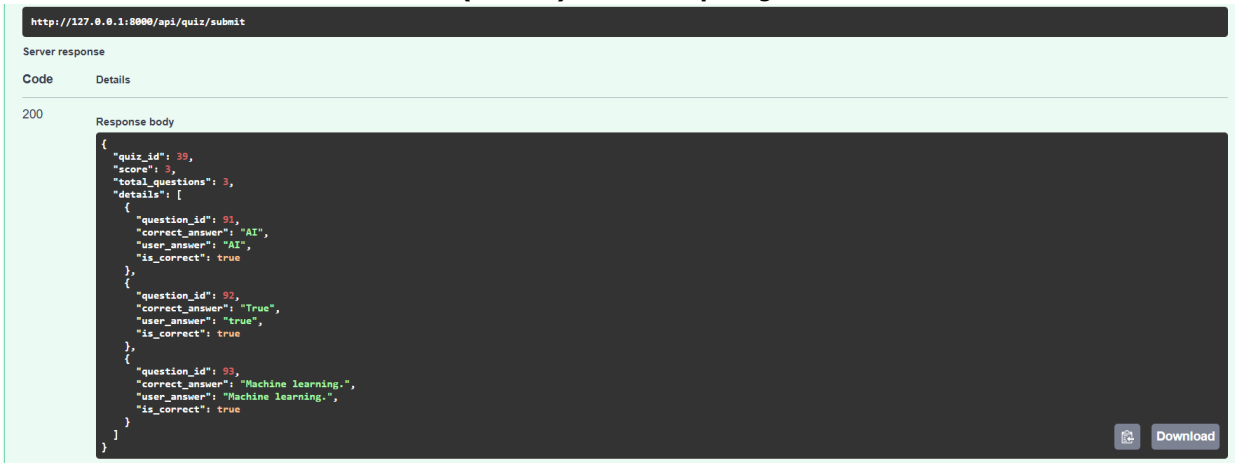


delete quiz

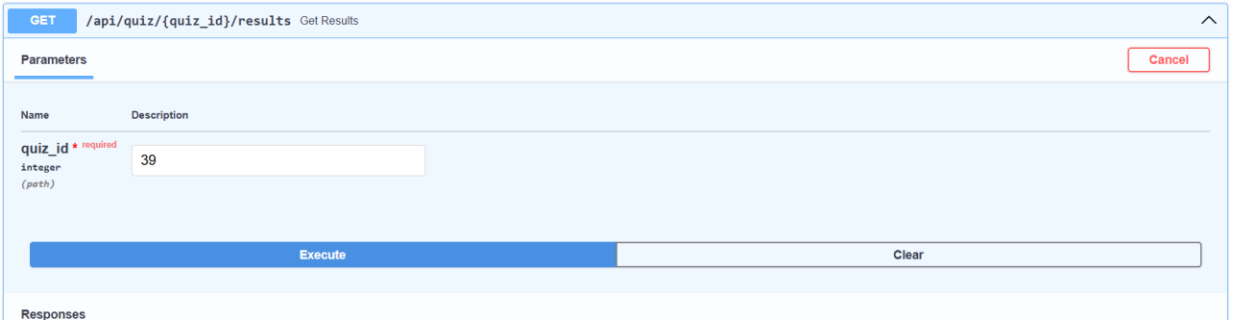


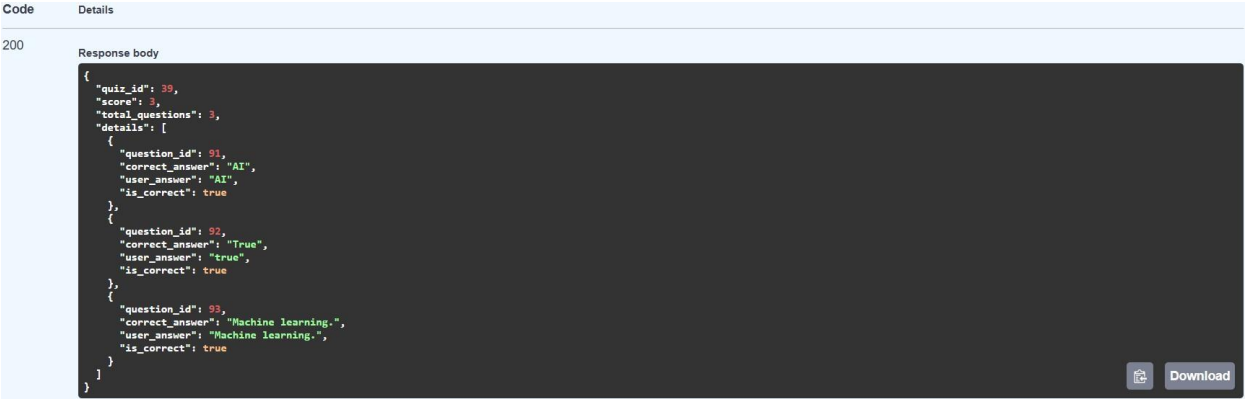
Submit Quiz





Get Results





Get History

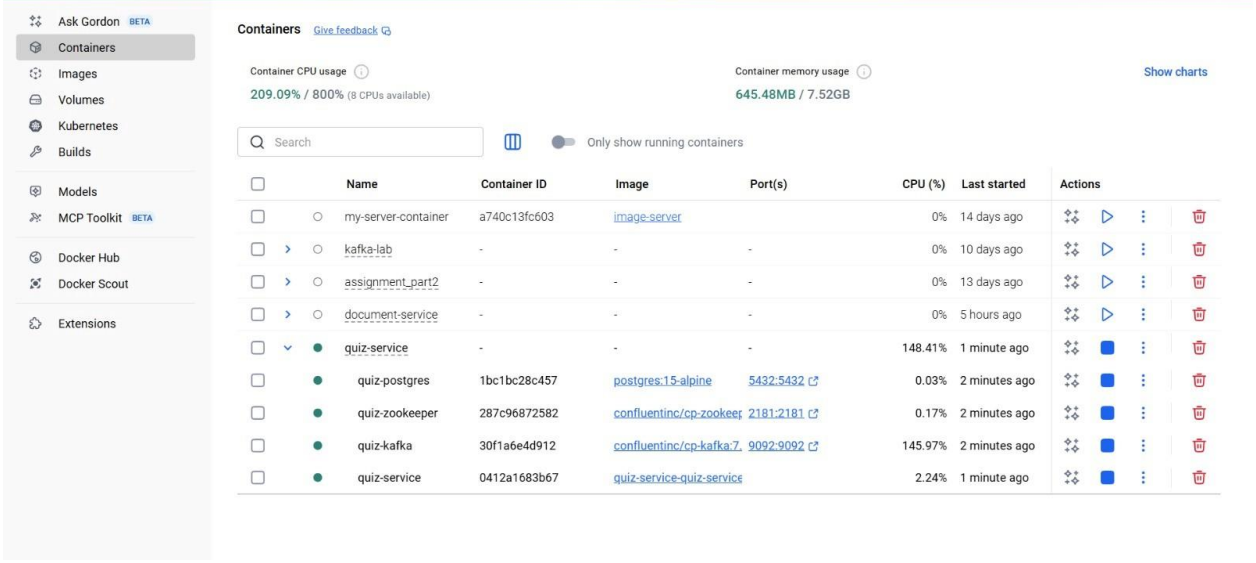
Docker is a platform that allows developers to package applications and dependencies into isolated containers, making it easy to develop, ship, and run applications in any environment. Implementing **Docker** for the **Quiz and Exercise Service** ensures a smooth and efficient deployment process.

Key Benefits of Docker Implementation:

1. **Isolation**: Each service (e.g., the web application, database, or caching system) can run in its container, ensuring that they do not interfere with each other.
2. **Consistency**: Docker ensures that the application runs in the same environment across different machines, preventing environment-specific bugs.
3. **Scalability**: Docker allows for easy scaling of services. For instance, if the quiz service experiences high traffic, more containers can be spun up to handle the load.
4. **Efficiency**: Docker containers are lightweight, so they consume fewer resources compared to traditional virtual machines.

Docker Compose for Multi-Container Setup:
To manage the various services such as the web application, database, and message brokers, **Docker Compose** is used. This tool helps define and run multi-container Docker applications.

- **docker-compose.yml** defines all the services that need to run in containers.
- **FastAPI Service**: The main service running the quiz generation logic.
- **PostgreSQL Database**: Stores quiz data, questions, and user responses.
- **Kafka**: Used for message streaming (e.g., real-time quiz generation).



6. Containerization Requirements

6.1 Docker Implementation

6.2 Container Orchestration

6.3 Container Registry

6. Containerization and Orchestration Requirements

This section documents the implementation of containerization strategies using Docker and the orchestration of microservices using Kubernetes (K8s) manifests. The architecture ensures scalability, self-healing, and efficient resource management.

6.3 Kubernetes Orchestration

The production environment utilizes Kubernetes manifests to define the desired state of the system, covering deployments, networking, auto-scaling, and storage.

A. Workload Management (Deployments)

The Deployment manifests define the application workloads, including replica counts, resource quotas (CPU/Memory), and health checks (Liveness/Readiness probes).

- **api-gateway-deployment.yaml**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: api-gateway-deployment
  labels:
    app: api-gateway
spec:

  replicas: 3
  strategy:
    type: RollingUpdate
  selector:
    matchLabels:
      app: api-gateway
  template:
    metadata:
      labels:
        app: api-gateway
    spec:
      containers:
      - name: api-gateway-container

        image: 234690396635.dkr.ecr.us-east-1.amazonaws.com/api-gateway-repo:latest
        ports:
        - containerPort: 8080


        resources:
          requests:
            cpu: "500m"
            memory: "1Gi"
          limits:
            cpu: "500m"
            memory: "1Gi"

        livenessProbe:
          httpGet:
            path: /health
```

```
        port: 8080
      initialDelaySeconds: 15
      timeoutSeconds: 5
    readinessProbe:
      httpGet:
        path: /ready
        port: 8080
      initialDelaySeconds: 5
      timeoutSeconds: 3
```

- **chat-service-deployment.yaml**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: chat-service-deployment
  labels:
    app: chat-service
spec:
  replicas: 2
  strategy:
    type: RollingUpdate
  selector:
    matchLabels:
      app: chat-service
  template:
    metadata:
      labels:
        app: chat-service
    spec:
      containers:
      - name: chat-service-container
        image: 234690396635.dkr.ecr.us-east-1.amazonaws.com/chat-service-repo:latest
        ports:
        - containerPort: 8080
        resources:
          requests:
            cpu: "2"
            memory: "4Gi"
          limits:
            cpu: "2"
            memory: "4Gi"
        livenessProbe:
          httpGet:
            path: /health
            port: 8080
          initialDelaySeconds: 15
          timeoutSeconds: 5
        readinessProbe:
          httpGet:
            path: /ready
            port: 8080
          initialDelaySeconds: 5
          timeoutSeconds: 3
```

```
        volumeMounts:
        - name: db-persistent-storage
          mountPath: /app/data/db
      volumes:
      - name: db-persistent-storage
        persistentVolumeClaim:
          claimName: chat-service-pvc
```

- **quiz-service-deployment.yaml**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: quiz-service-deployment
  labels:
    app: quiz-service
spec:
  replicas: 2
  strategy:
    type: RollingUpdate
  selector:
    matchLabels:
      app: quiz-service
  template:
    metadata:
      labels:
        app: quiz-service
    spec:
      containers:
      - name: quiz-service-container
        image: 234690396635.dkr.ecr.us-east-1.amazonaws.com/quiz-service-repo:latest
        ports:
        - containerPort: 8080
        resources:
          requests:
            cpu: "1"
            memory: "2Gi"
          limits:
            cpu: "1"
            memory: "2Gi"
        livenessProbe:
          httpGet:
            path: /health
            port: 8080
          initialDelaySeconds: 15
          timeoutSeconds: 5
        readinessProbe:
          httpGet:
            path: /ready
            port: 8080
          initialDelaySeconds: 5
          timeoutSeconds: 3
        volumeMounts:
        - name: db-persistent-storage
```

```
        mountPath: /app/data/quiz


      volumes:
      - name: db-persistent-storage
        persistentVolumeClaim:
          claimName:  quiz-service-pvc
```

- **document-reader-deployment.yaml**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: document-reader-deployment
  labels:
    app: document-reader-service
spec:
  replicas: 2
  strategy:
    type: RollingUpdate
  selector:
    matchLabels:
      app: document-reader-service
  template:
    metadata:
      labels:
        app: document-reader-service
    spec:
      containers:
      - name:  document-reader-container
        image: 234690396635.dkr.ecr.us-east-1.amazonaws.com/document-reader-repo:latest
        ports:
        - containerPort: 8080
        resources:
          requests:
            cpu: "1"
            memory: "2Gi"
          limits:
            cpu: "1"
            memory: "2Gi"
        livenessProbe:
          httpGet:
            path: /health
            port: 8080
          initialDelaySeconds: 15
          timeoutSeconds: 5
        readinessProbe:
          httpGet:
            path: /ready
            port: 8080
          initialDelaySeconds: 5
          timeoutSeconds: 3
        volumeMounts:
        - name: db-persistent-storage
```

```
        mountPath: /app/data/docs
      volumes:
      - name: db-persistent-storage
        persistentVolumeClaim:
          claimName:  document-reader-pvc
```

- **stt-service-deployment.yaml**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: stt-service-deployment
  labels:
    app: stt-service
spec:
  replicas: 2
  strategy:
    type: RollingUpdate
  selector:
    matchLabels:
      app: stt-service
  template:
    metadata:
      labels:
        app: stt-service
    spec:
      containers:
      - name:  stt-service-container
        image: 234690396635.dkr.ecr.us-east-1.amazonaws.com/stt-service-repo:latest
        ports:
        - containerPort: 8080
        resources:
          requests:
            cpu: "2"
            memory: "4Gi"
          limits:
            cpu: "2"
            memory: "4Gi"
        livenessProbe:
          httpGet:
            path: /health
            port: 8080
          initialDelaySeconds: 15
          timeoutSeconds: 5
        readinessProbe:
          httpGet:
            path: /ready
            port: 8080
          initialDelaySeconds: 5
          timeoutSeconds: 3
```

- **tts-service-deployment.yaml**

```
apiVersion: apps/v1
```

```
kind: Deployment
metadata:
  name: tts-service-deployment
  labels:
    app: tts-service
spec:
  replicas: 2
  strategy:
    type: RollingUpdate
  selector:
    matchLabels:
      app: tts-service
  template:
    metadata:
      labels:
        app: tts-service
    spec:
      containers:
      - name: tts-service-container

        image: 234690396635.dkr.ecr.us-east-1.amazonaws.com/tts-service-repo:latest
        ports:
        - containerPort: 8080


        resources:
          requests:
            cpu: "1"
            memory: "2Gi"
          limits:
            cpu: "1"
            memory: "2Gi"
        livenessProbe:
          httpGet:
            path: /health
            port: 8080
          initialDelaySeconds: 15
          timeoutSeconds: 5
        readinessProbe:
          httpGet:
            path: /ready
            port: 8080
          initialDelaySeconds: 5
          timeoutSeconds: 3
```

B. Service Discovery & Networking

Kubernetes Services are configured to expose pods internally or externally.

- Internal Communication: The document-reader-service uses ClusterIP for secure internal communication.
- External Access: The api-gateway-service uses LoadBalancer to expose the application to the internet.

  - **document-reader-service.yaml**

```
apiVersion: v1
kind: Service
metadata:
  name: document-reader-service
  labels:
    app: document-reader-service
spec:
  type: ClusterIP
  selector:
    app: document-reader-service
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
```

- **quiz-service-service.yaml**

```
apiVersion: v1
kind: Service
metadata:
  name: quiz-service
  labels:
    app: quiz-service
spec:
  type: ClusterIP
  selector:
    app: quiz-service
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
```

- **api-gateway-service.yaml**

```
apiVersion: v1
kind: Service
metadata:
  name: api-gateway-service
  labels:
    app: api-gateway
spec:

  type: LoadBalancer
  selector:
    app: api-gateway
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
```

- **stt-service-service.yaml**

```
apiVersion: v1
kind: Service
```

```
metadata:
  name: stt-service
  labels:
    app: stt-service
spec:

  type: ClusterIP
  selector:

    app: stt-service
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
```

- **tts-service-service.yaml**

```
apiVersion: v1
kind: Service
metadata:
  name: tts-service
  labels:
    app: tts-service
spec:
  type: ClusterIP
  selector:
    app: tts-service
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
```

- **chat-service-service.yaml**

```
apiVersion: v1
kind: Service
metadata:
  name: chat-service
  labels:
    app: chat-service
spec:
  type: ClusterIP
  selector:
    app: chat-service
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
```

- **Auto-scaling Strategies (HPA)**

Horizontal Pod Autoscalers (HPA) are implemented to automatically adjust the number of pods based on CPU utilization. This ensures the system handles traffic spikes efficiently.

- **chat-service-hpa.yaml**

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: chat-service-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: chat-service-deployment
  minReplicas: 2
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 75
```

- **document-reader-hpa.yaml**

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: document-reader-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: document-reader-deployment
  minReplicas: 2
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 75
```

- **tts-service-hpa.yaml**

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: tts-service-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: tts-service-deployment
  minReplicas: 2
  maxReplicas: 10
```

```
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 75
```

- **quiz-service-hpa.yaml**

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: quiz-service-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: quiz-service-deployment
  minReplicas: 2
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 75
```

- **stt-service-hpa.yaml**

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: stt-service-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: stt-service-deployment
  minReplicas: 2
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 75
```

- **api-gateway-hpa.yaml**

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
```

```
metadata:
  name: api-gateway-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: api-gateway-deployment
  minReplicas: 3
  maxReplicas: 15
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 75
```

- **Persistent Storage (State Management)**
  For services requiring data persistence (like the Chat Service which may cache vector data locally), Persistent Volume Claims (PVC) are used.
    - **chat-service-pvc.yaml**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: chat-service-pvc
spec:

  accessModes:
    - ReadWriteOnce

  resources:
    requests:
      storage: 5Gi
```

    - **document-reader-pvc.yaml**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: document-reader-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
  # storageClassName: "standard"
```

    - **quiz-service-pvc.yaml**

```
apiVersion: v1
kind: PersistentVolumeClaim
```

(CSE353) Cloud Computing

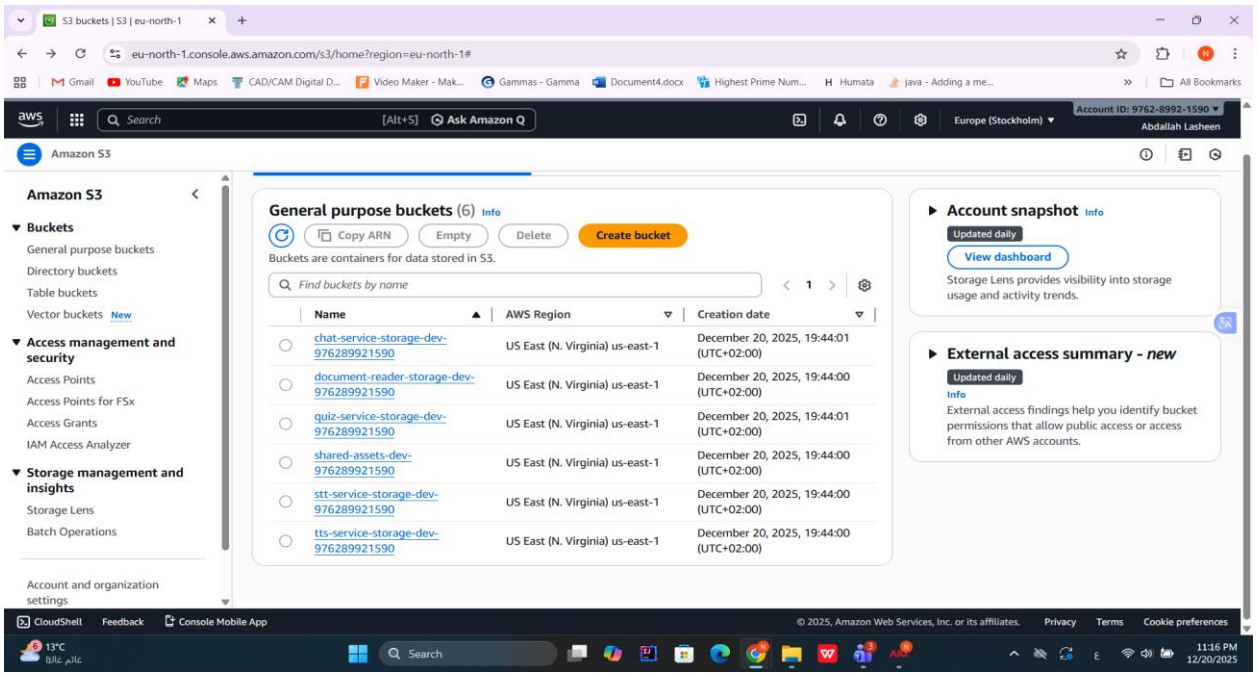
```
  metadata:
    name: quiz-
  service-pvc spec:
    accessModes:
      -
    ReadWrite
    Once
    resources:
      requests:
        storage: 3Gi
# storageClassName: "standard
```

Phase 3

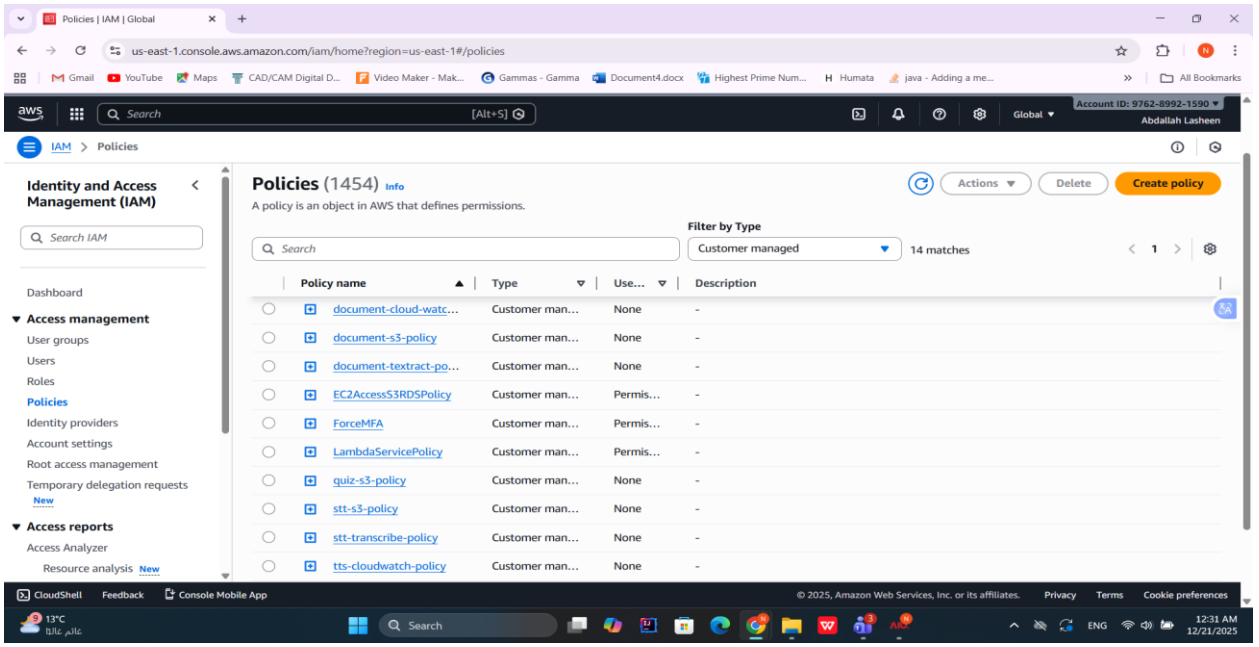
7.1 Storage Isolation Principles

S3 Isolation:

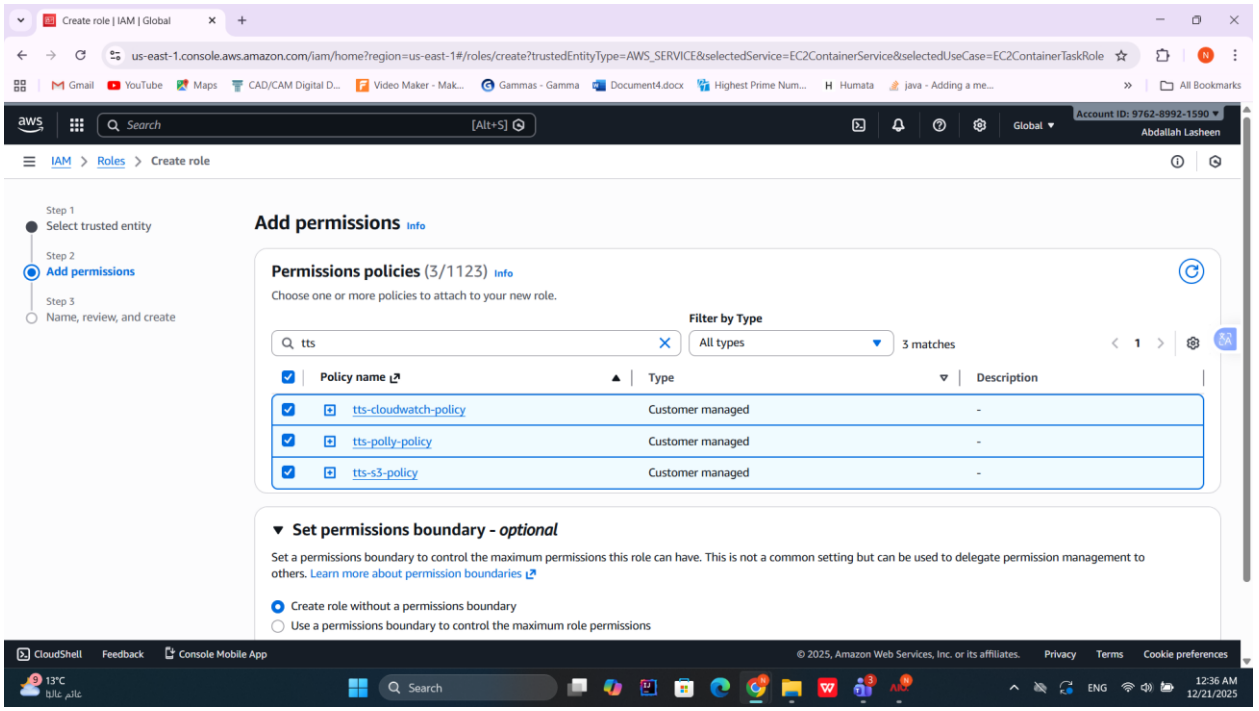
- Separate S3 bucket per service



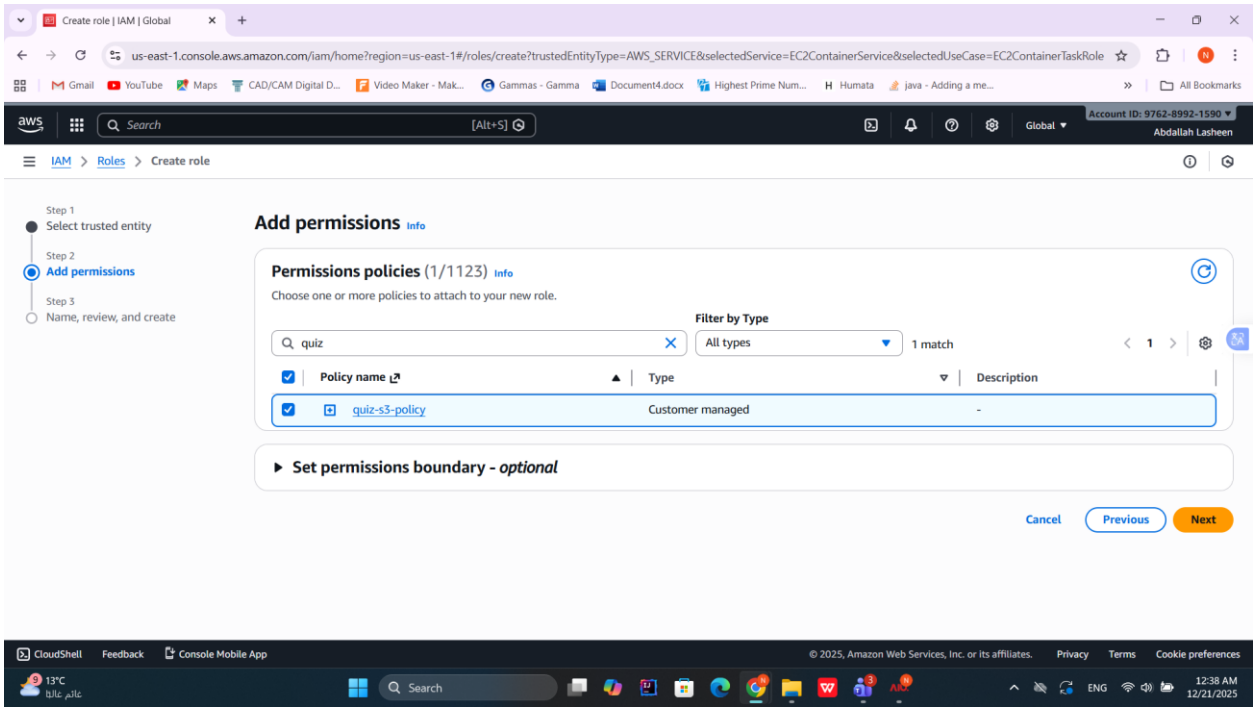
- Bucket policies enforce service-specific IAM role access only



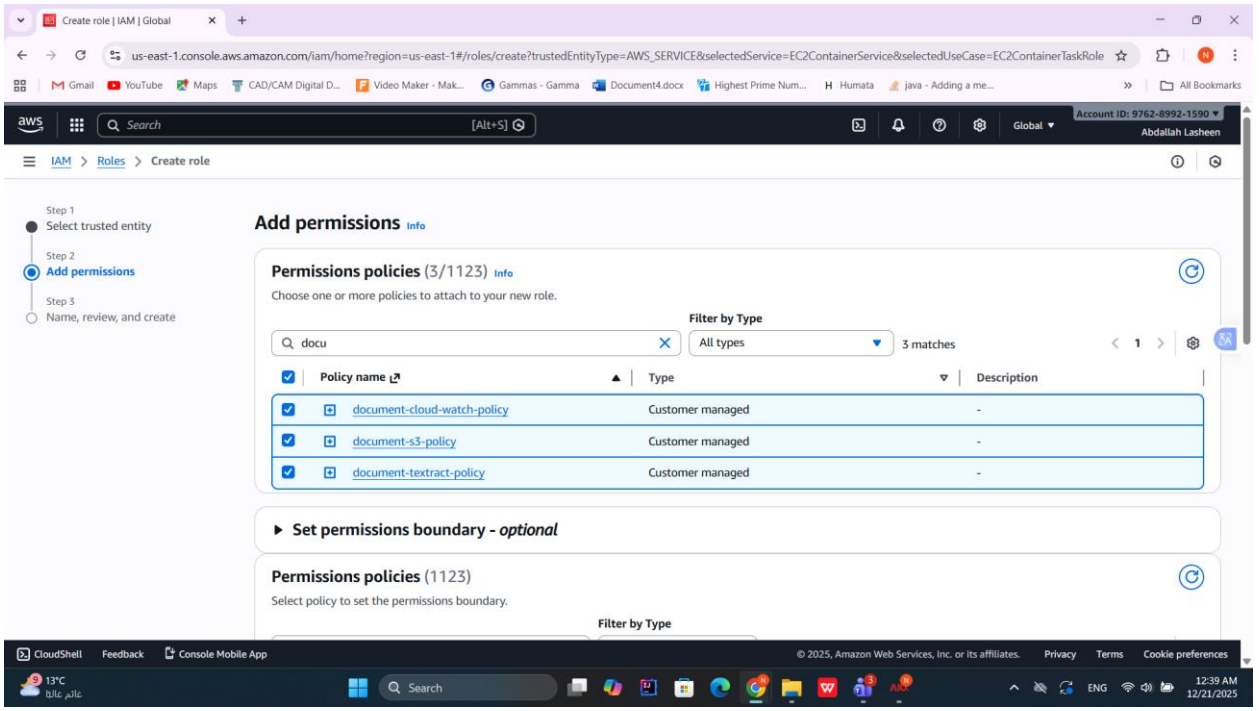
- Created IAM TTS Policies



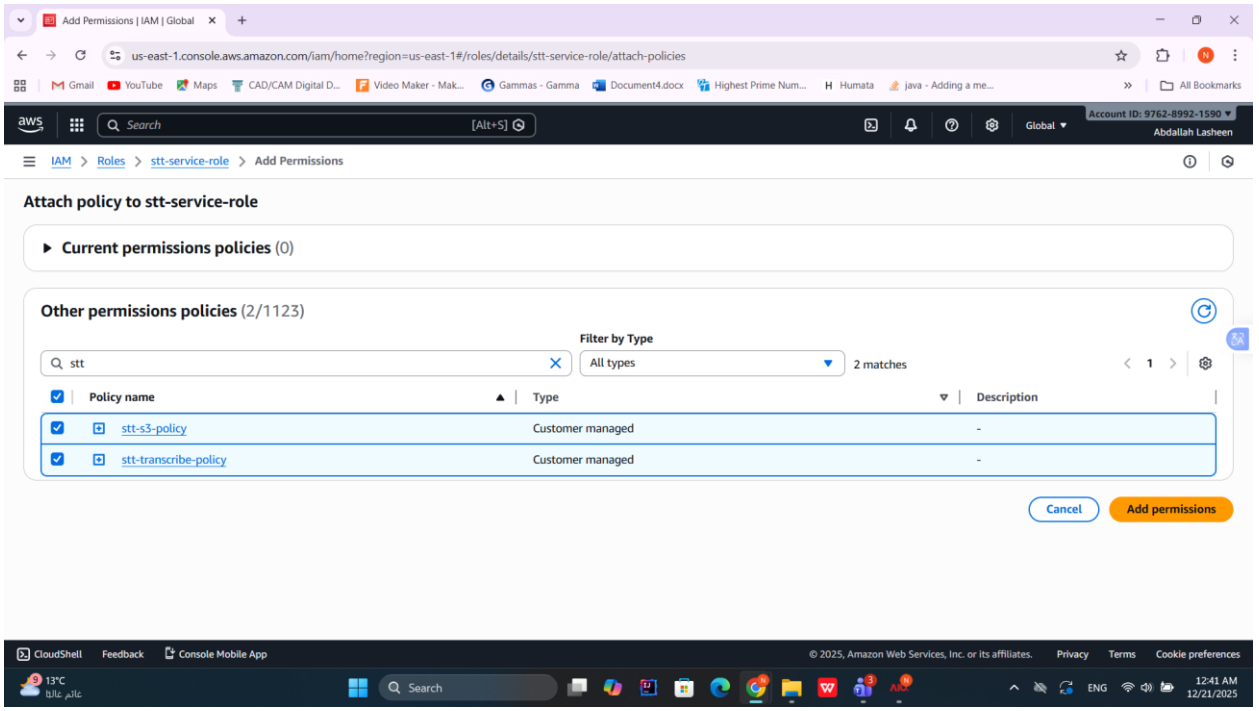
- Created IAM Quiz Policies



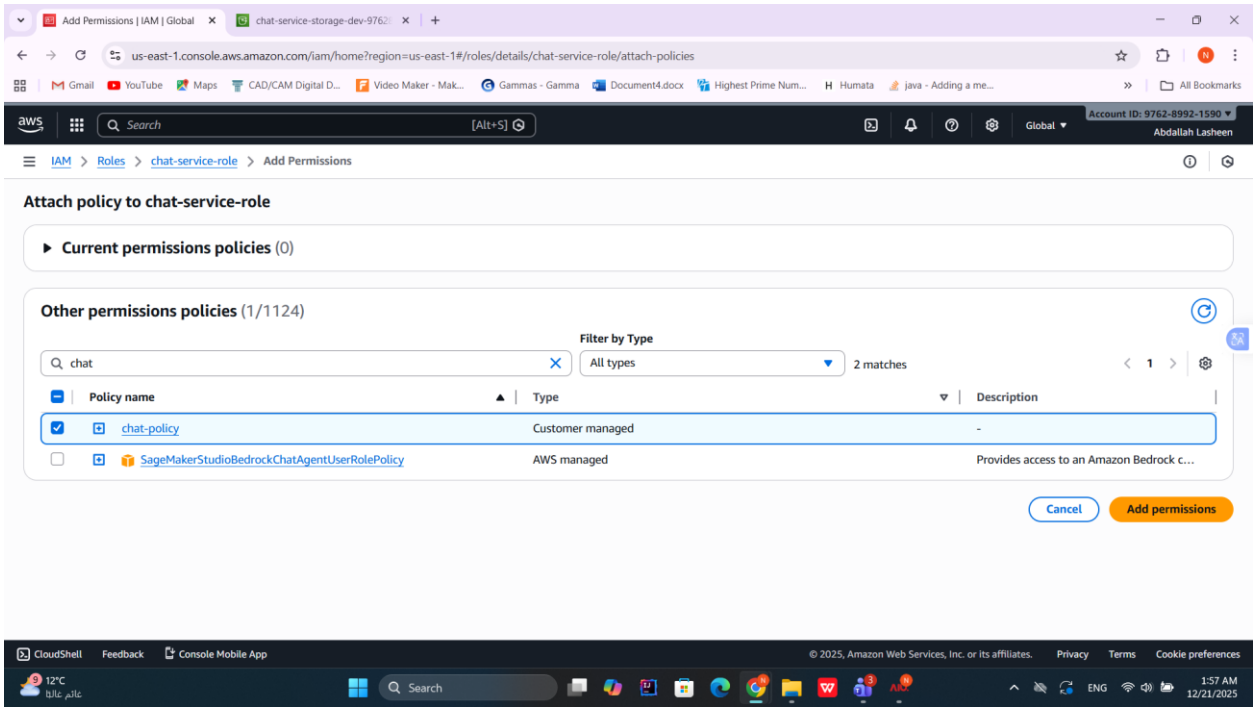
- Created IAM Document Policies



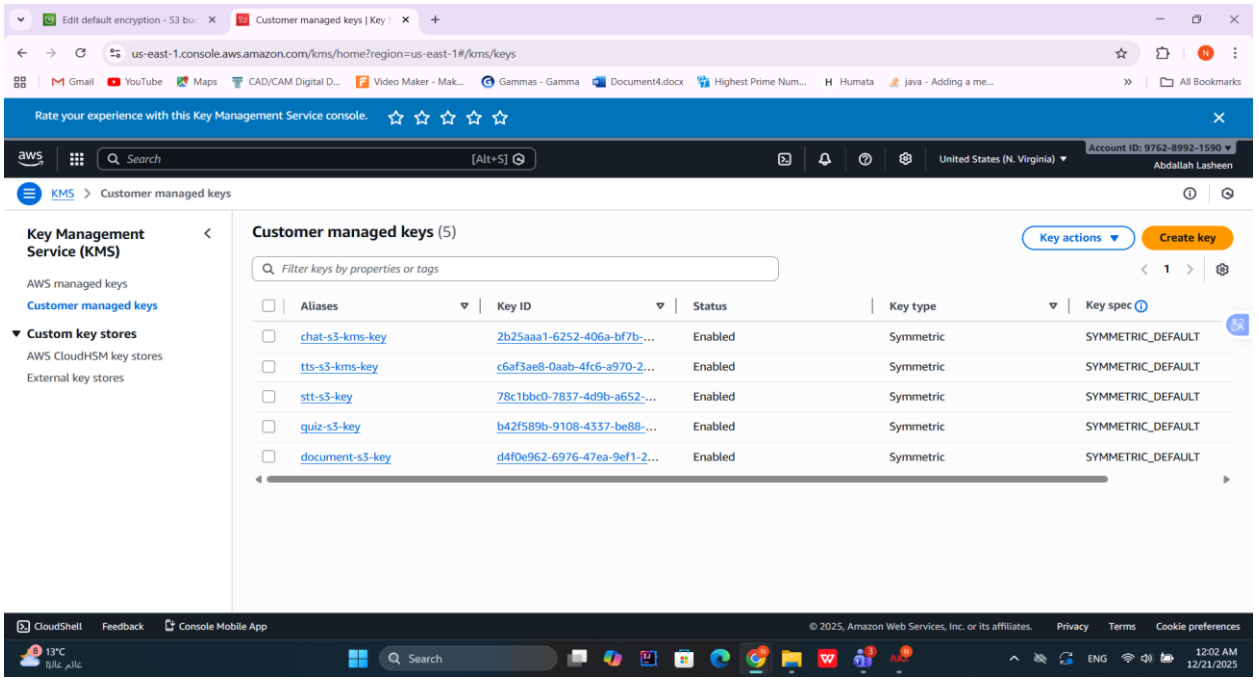
- Created IAM STT Policies



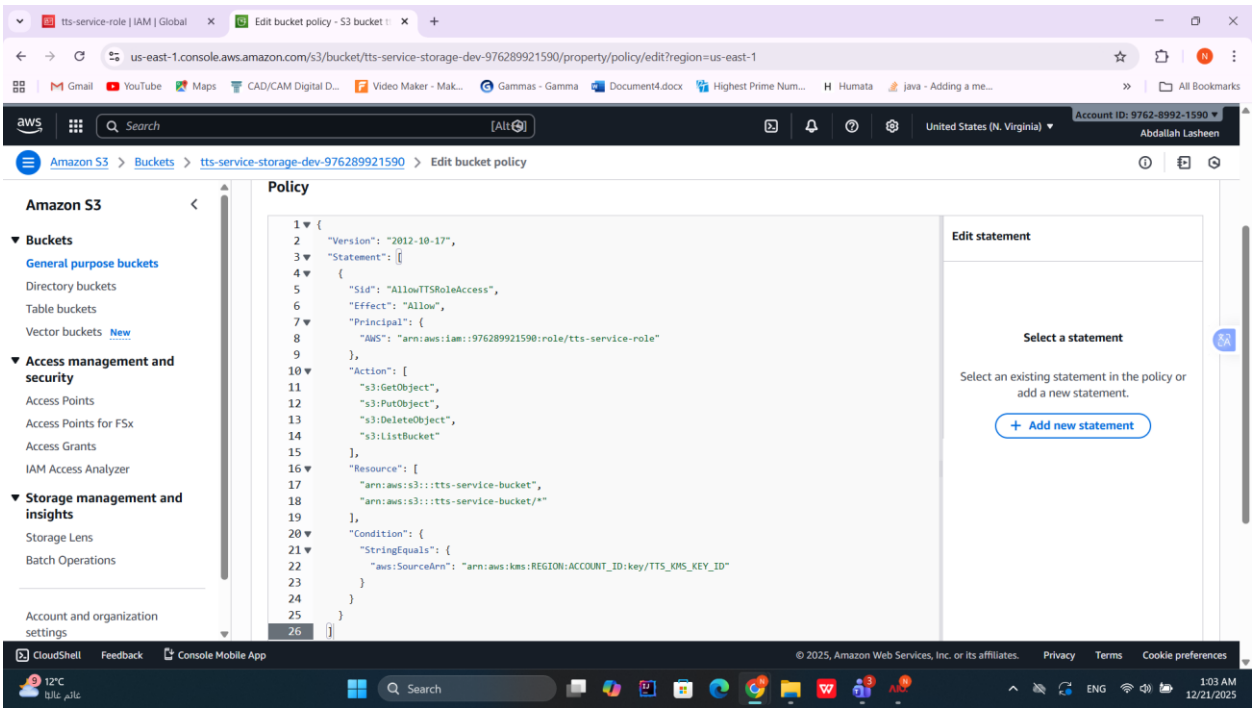
- Created IAM Chat Policies



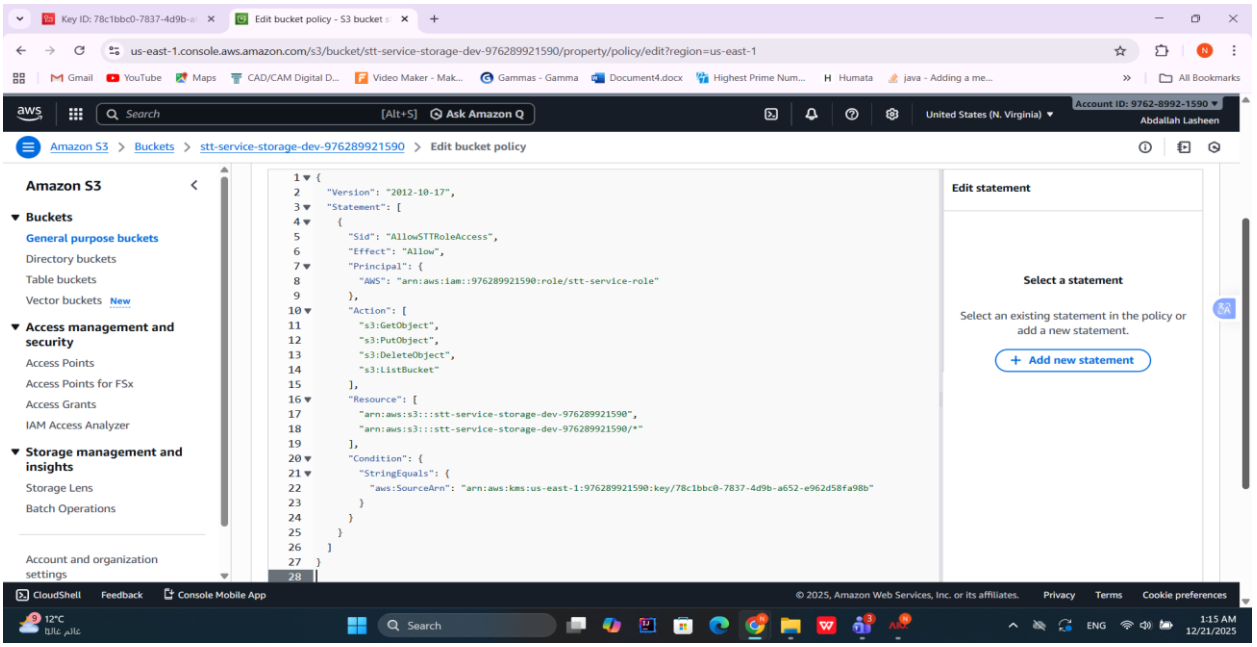
- Separate encryption keys per bucket (KMS)



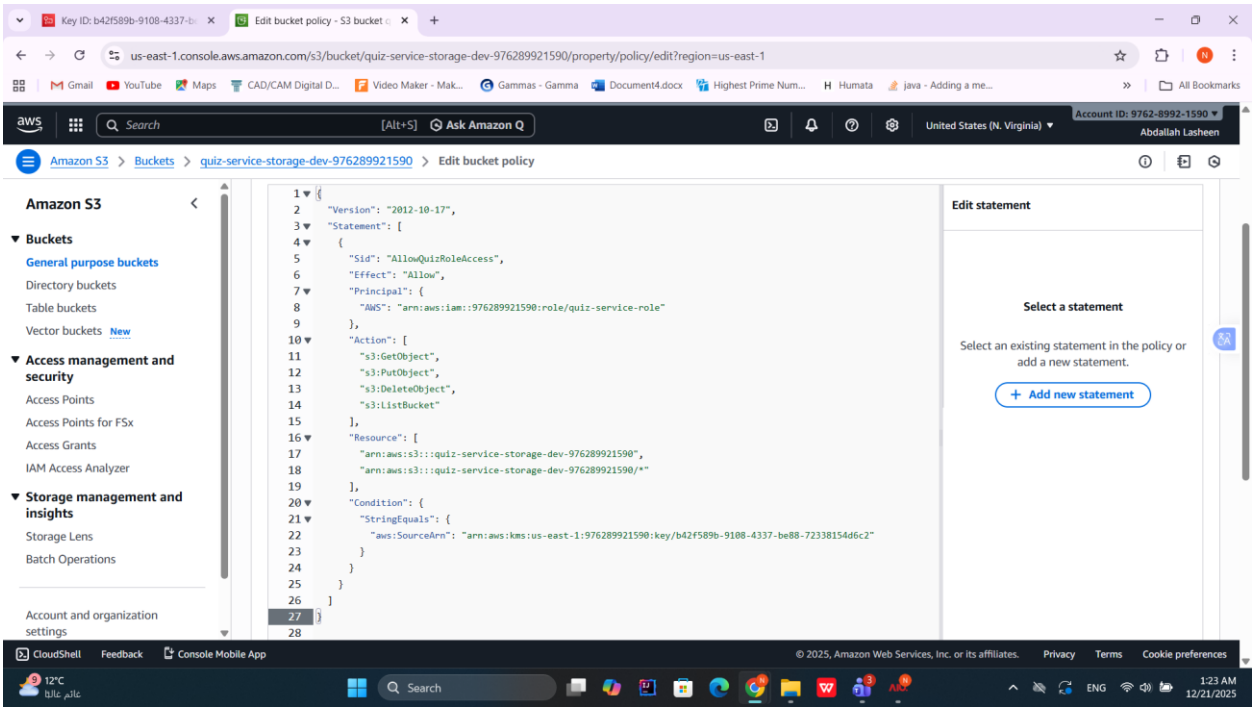
- TTS Bucket Policy



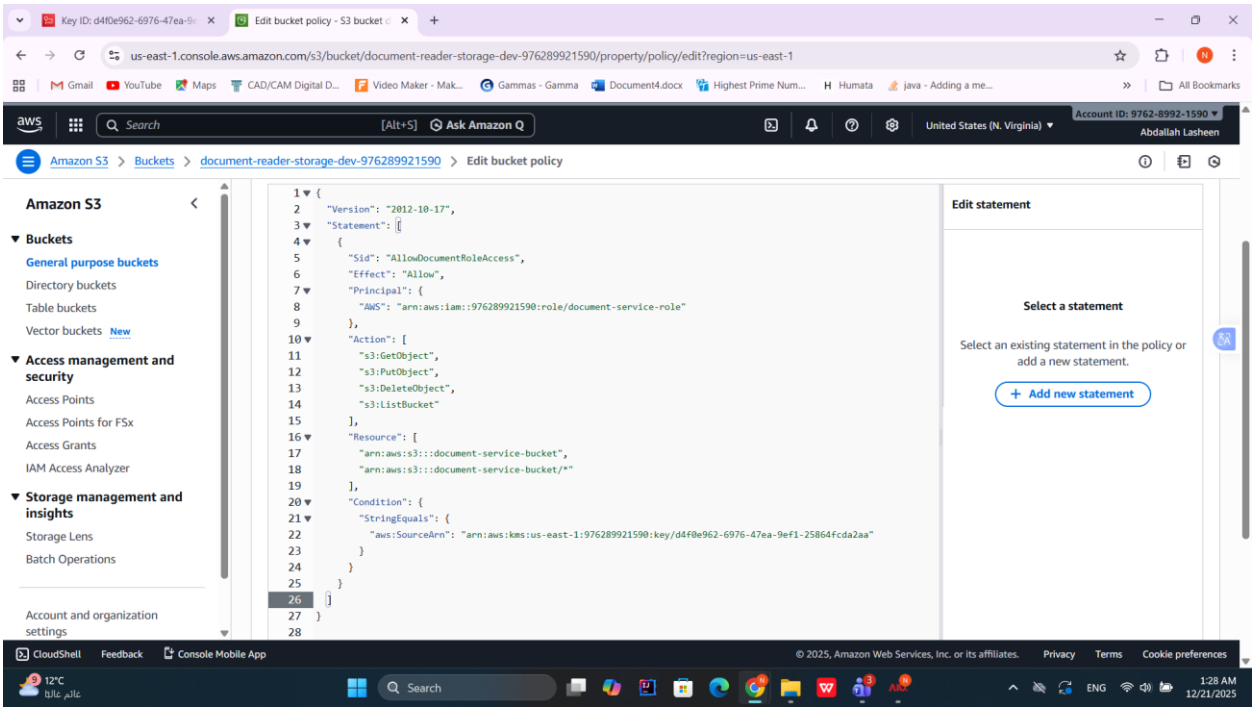
- STT Bucket Policy



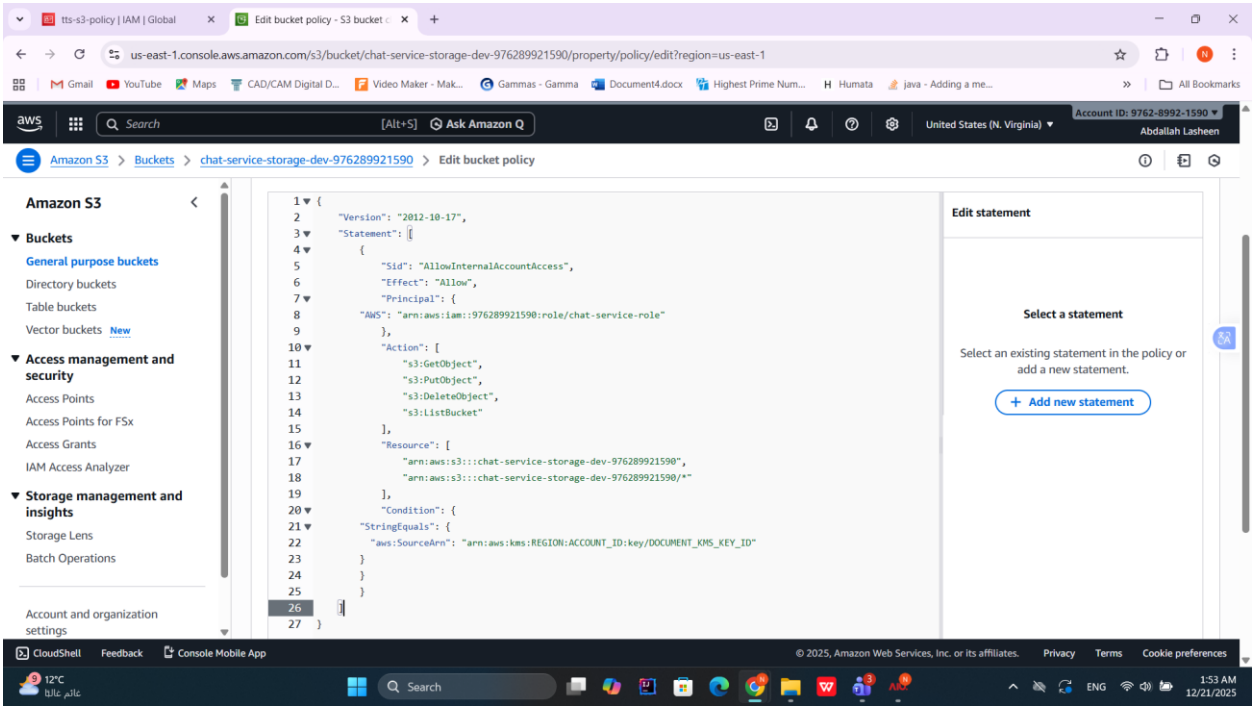
- Quiz Bucket Policy



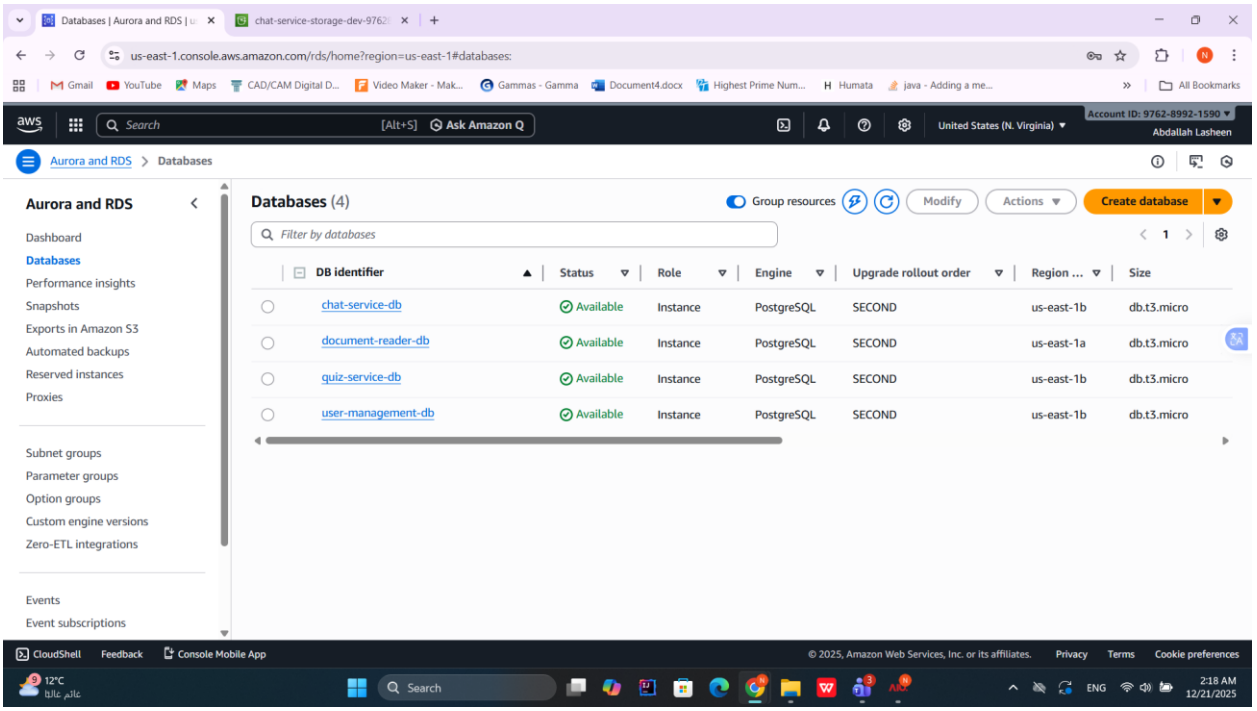
- Document Bucket Policy

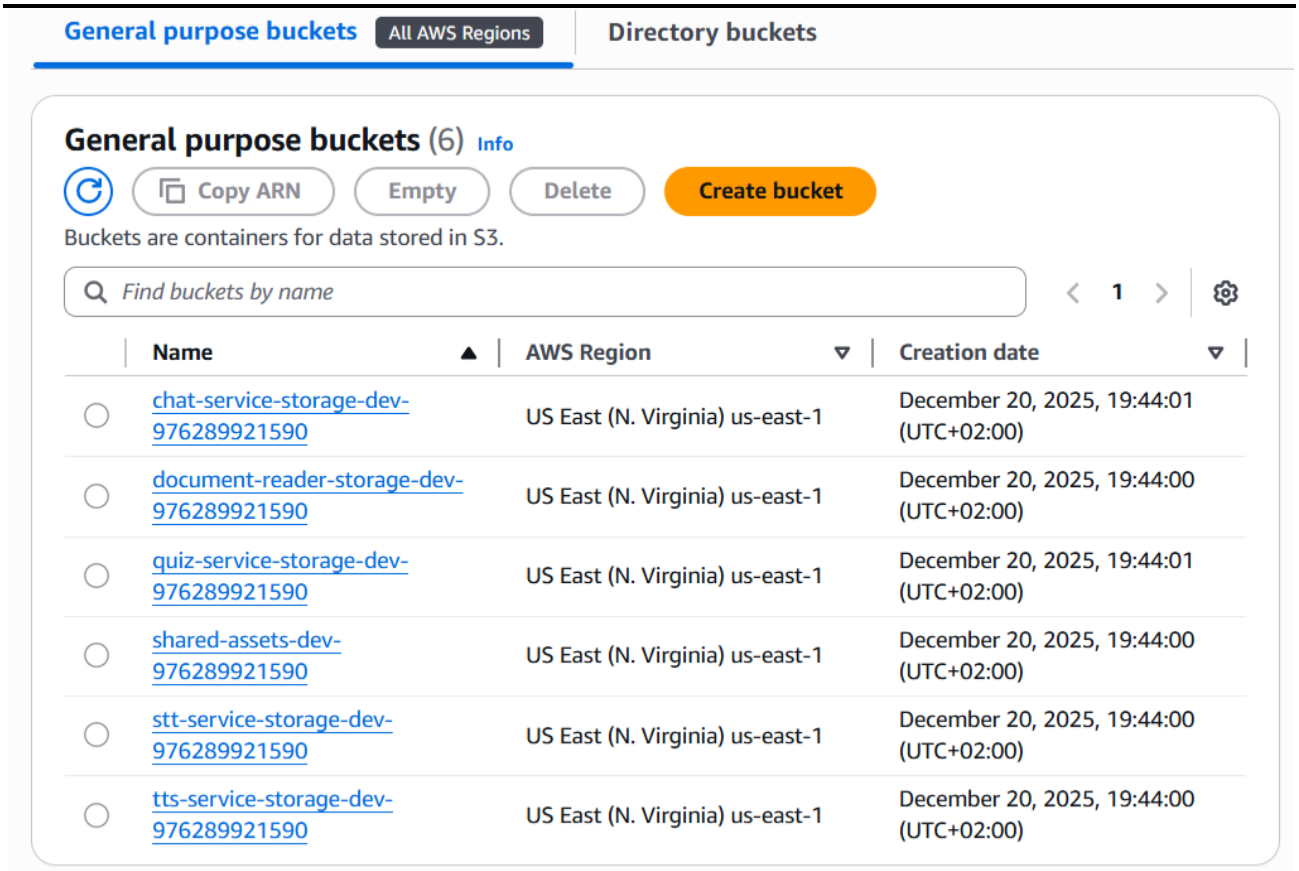


- Chat Bucket Policy

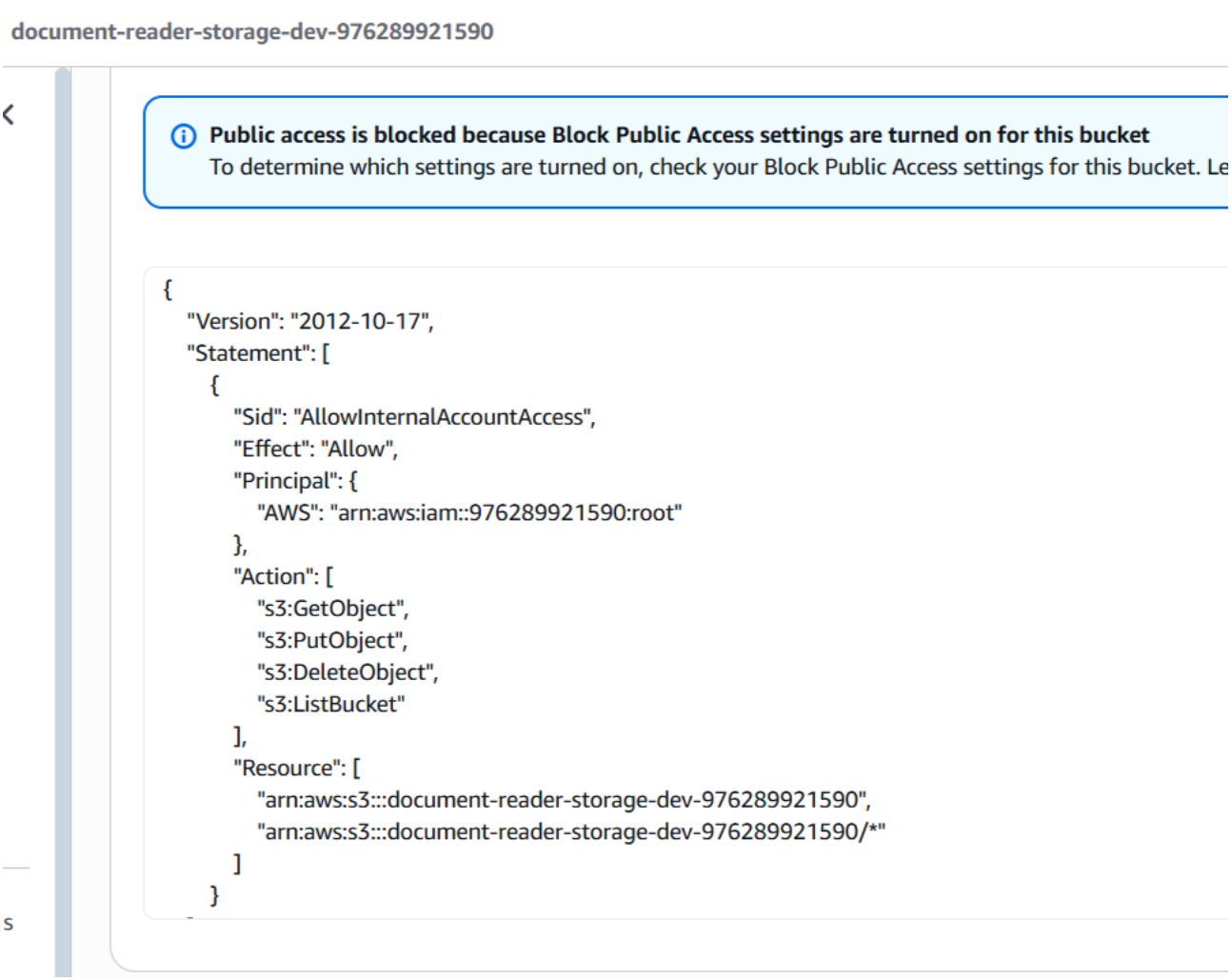


Database Isolation:

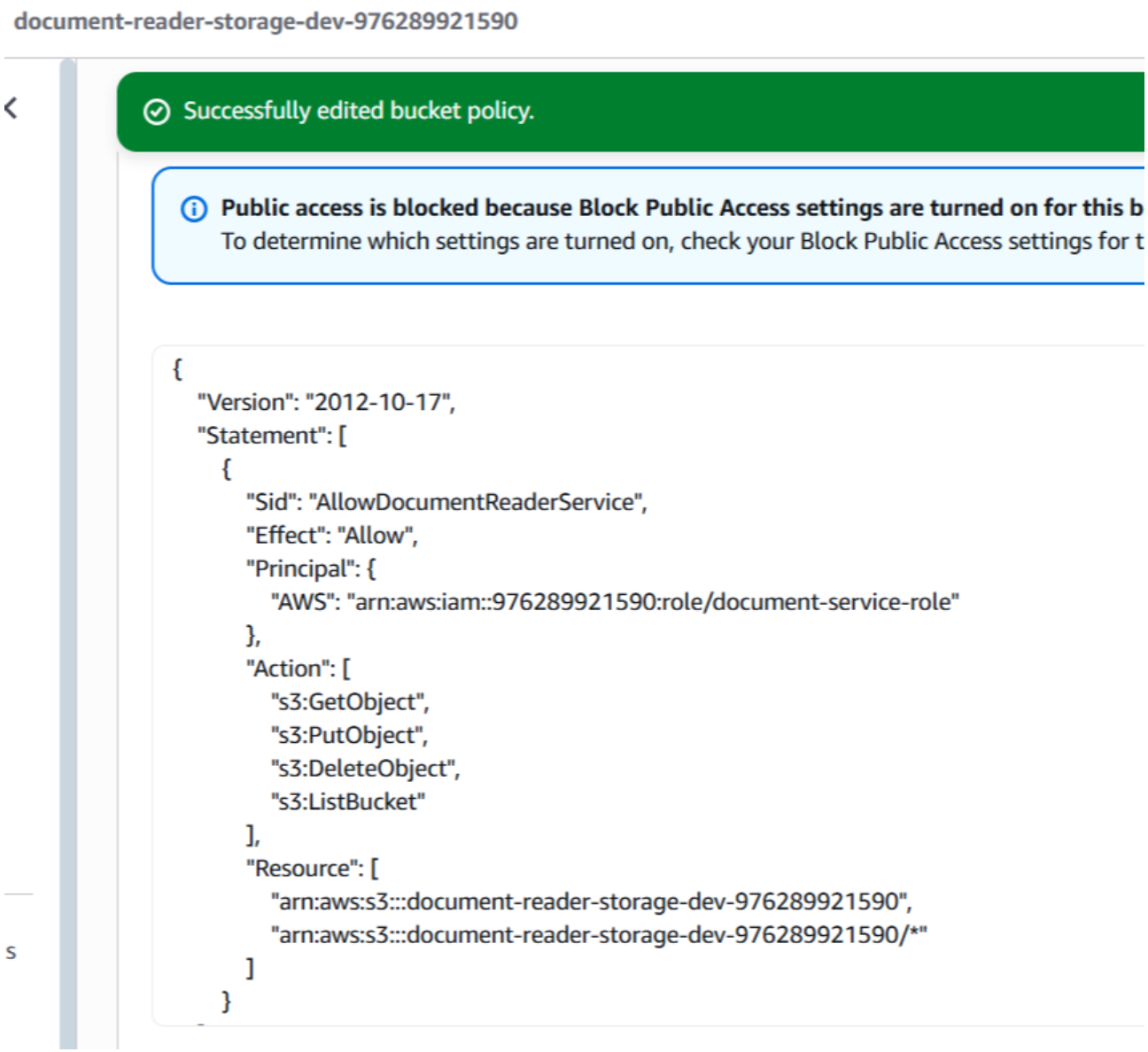




Each bucket is for different services like chat, document reader, quiz, shared assets, STT (speech-to-text), and TTS (text-to-speech)



This policy allows full internal access (get, put, delete, list) to the specified AWS account for this bucket and its contents, while public access is completely blocked



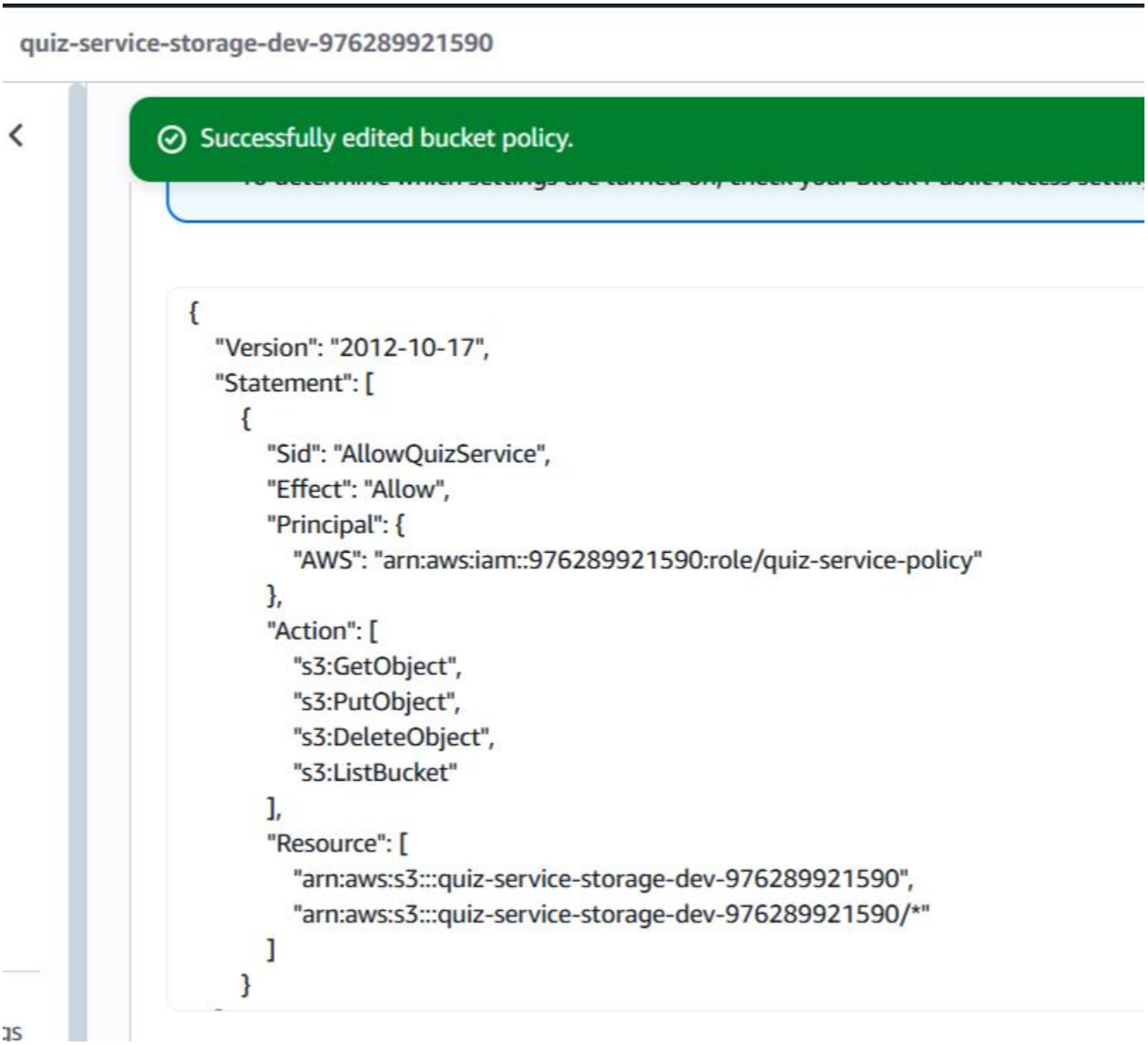
The bucket policy now grants full access (get, put, delete, list) to the **document-service-role** within the same AWS account, while public access remains blocked.

quiz-service-storage-dev-976289921590

Public access is blocked because Block Public Access settings are turned on fo
To determine which settings are turned on, check your Block Public Access settir

```
"Statement": [
  {
    "Sid": "AllowInternalAccountAccess",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::976289921590:root"
    },
    "Action": [
      "s3:GetObject",
      "s3:PutObject",
      "s3:DeleteObject",
      "s3:ListBucket"
    ],
    "Resource": [
      "arn:aws:s3:::quiz-service-storage-dev-976289921590",
      "arn:aws:s3:::quiz-service-storage-dev-976289921590/*"
    ]
  }
]
}
```

This policy grants full internal access (get, put, delete, list) to the root account for the quiz-service bucket, while public access remains blocked



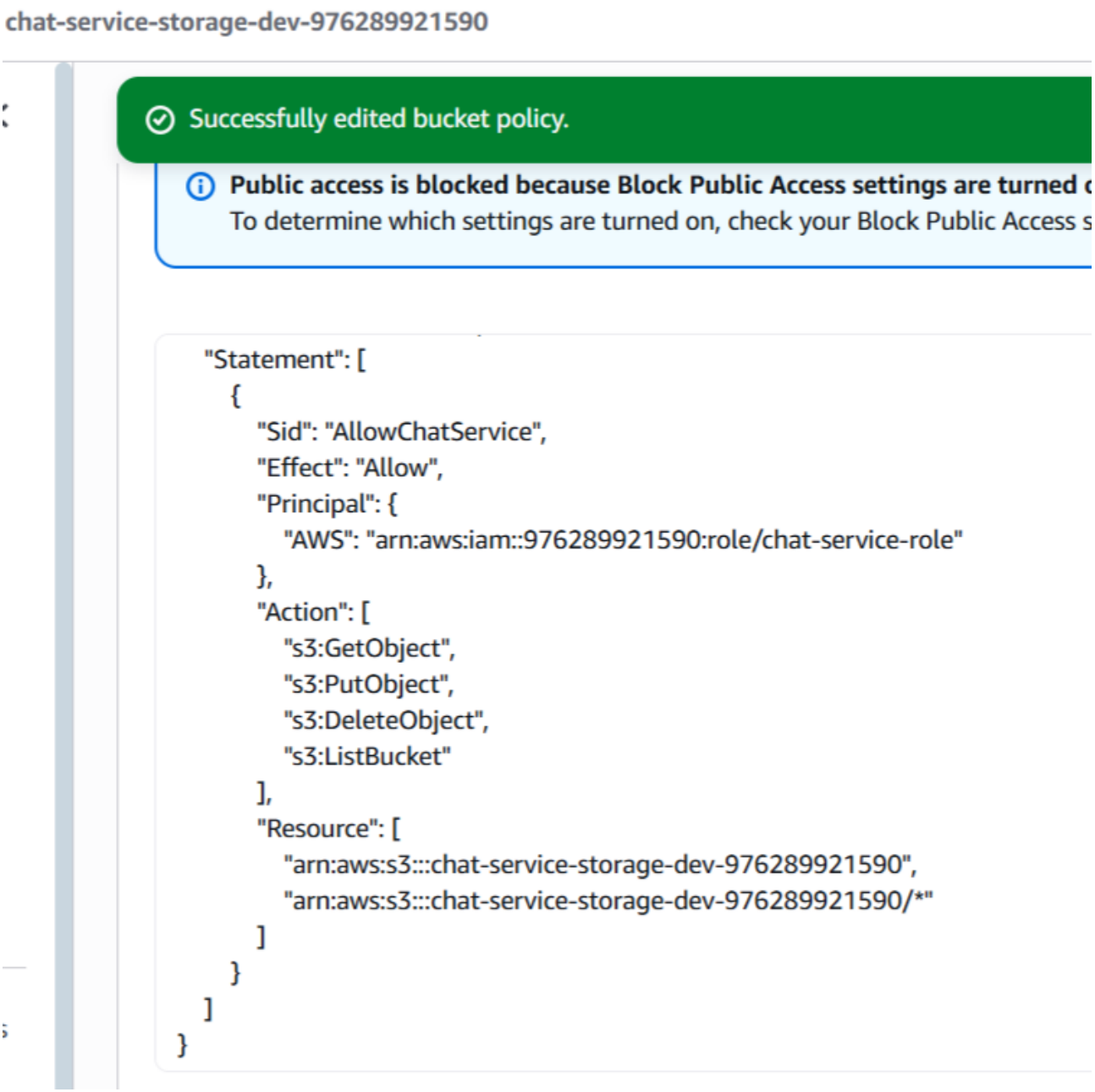
The bucket policy now grants full access (get, put, delete, list) to the **quiz-service IAM role**, replacing the previous root account access, while public access remains blocked

chat-service-storage-dev-976289921590

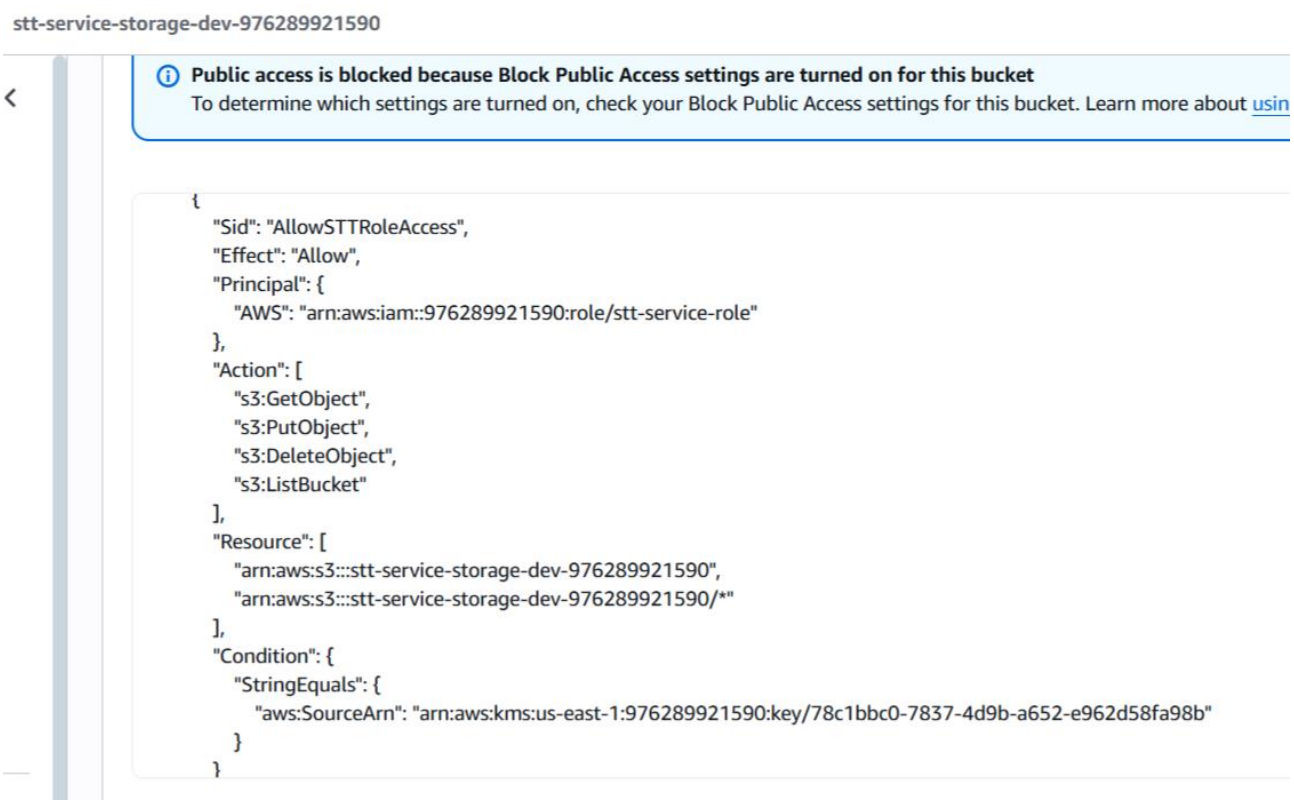
To determine which settings are turned on, check your Block Public Access setti

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowInternalAccountAccess",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::976289921590:role/chat-service-role"
      },
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::chat-service-storage-dev-976289921590",
        "arn:aws:s3:::chat-service-storage-dev-976289921590/*"
      ],
      "Condition": {
```

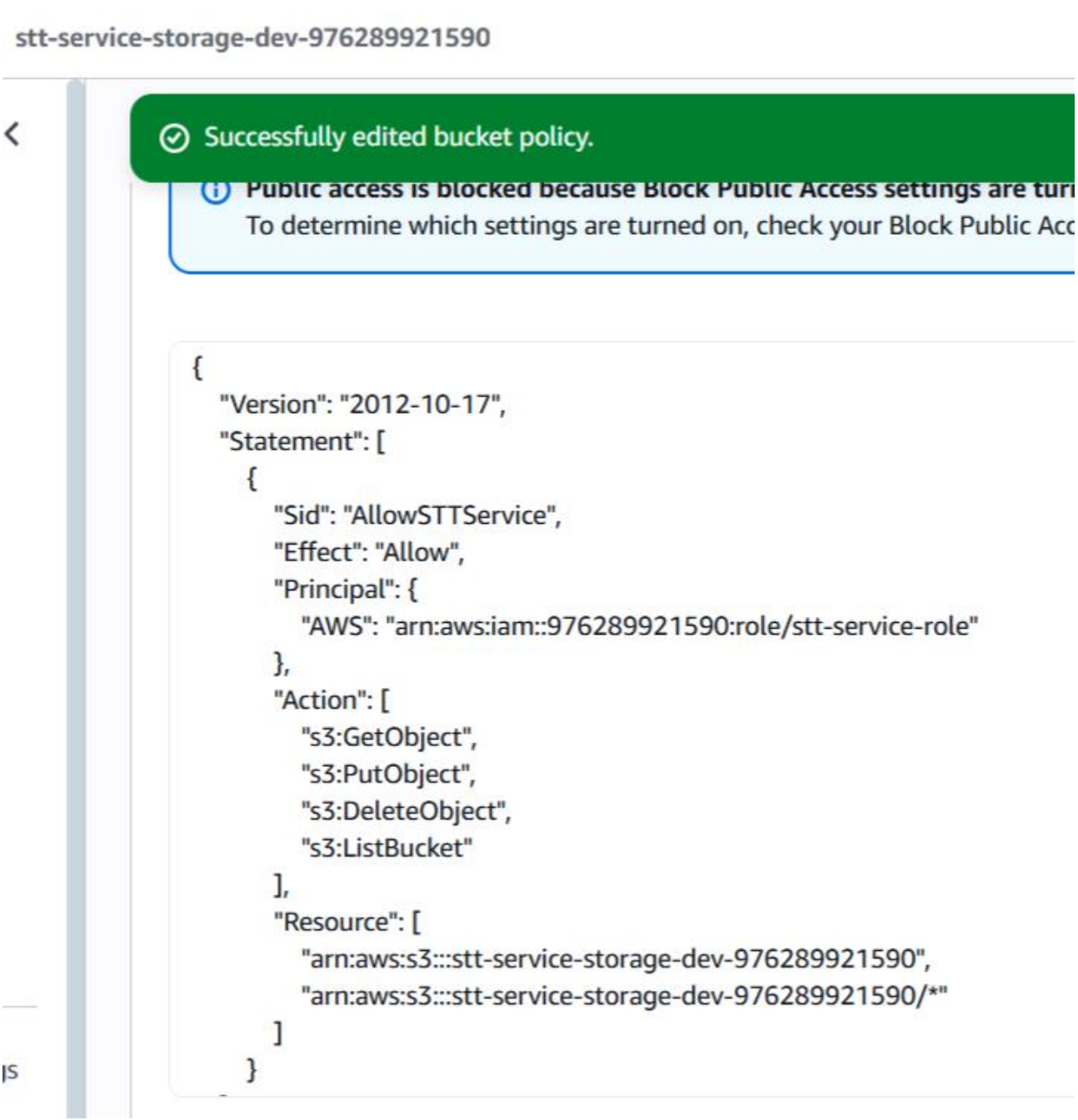
This policy grants full access (get, put, delete, list) to the **chat-service IAM role**, while public access remains blocked



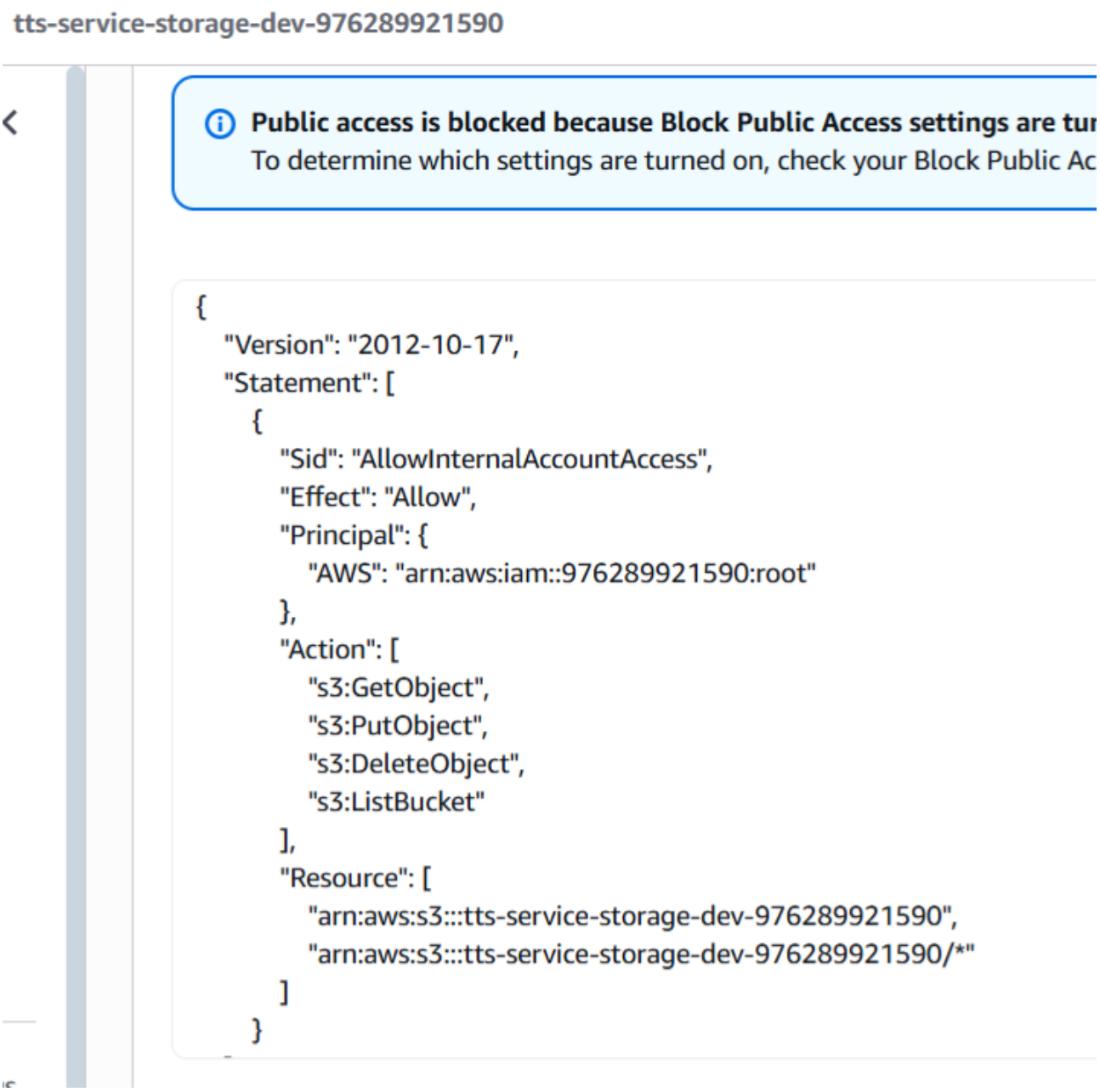
The bucket policy now grants full access (get, put, delete, list) to the **chat-service IAM role**, while public access remains blocked



This policy grants full access (get, put, delete, list) to the **STT service IAM role**, but only when requests originate from the specified KMS key, while public access remains blocked

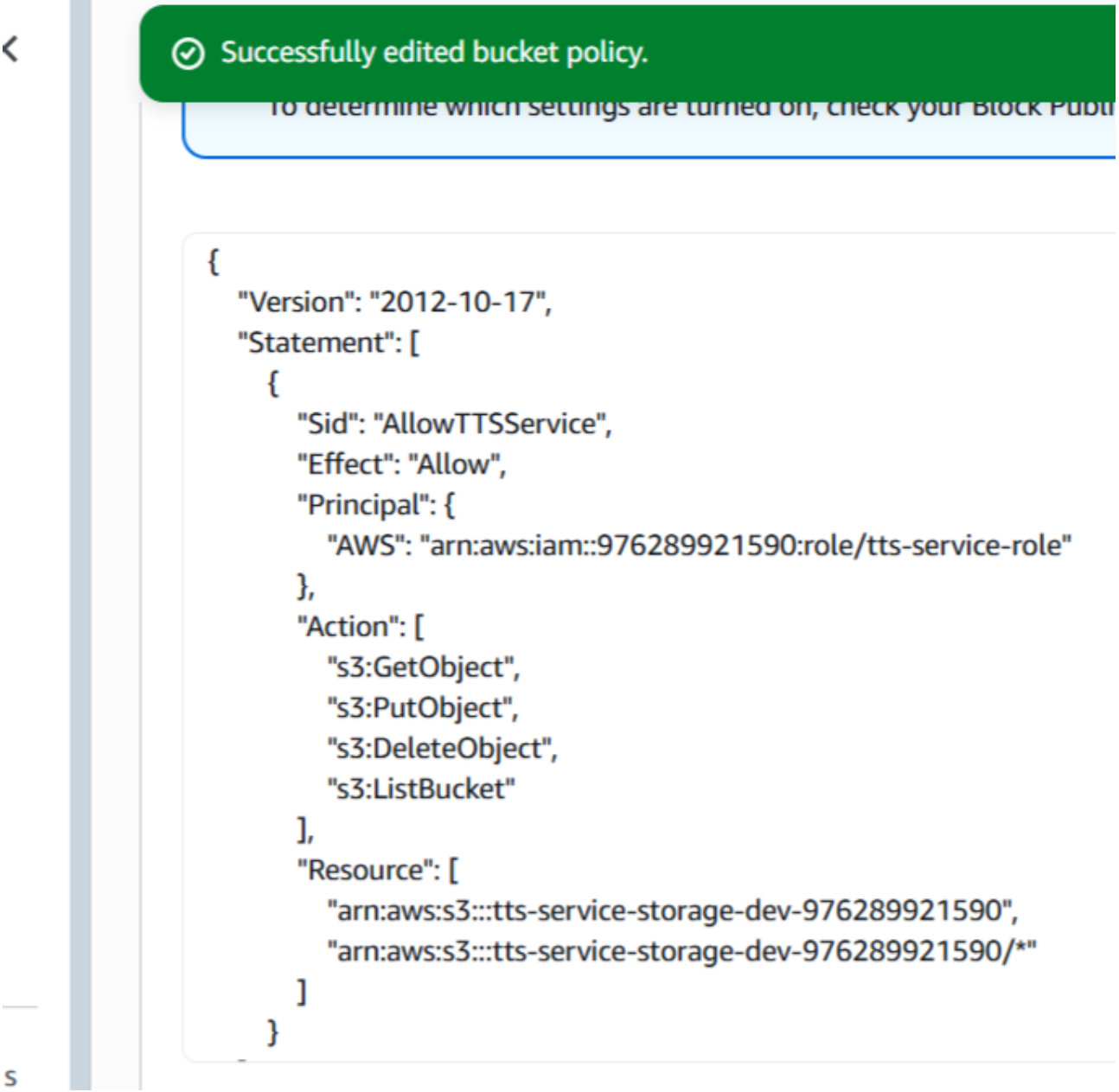


The bucket policy now grants full access (get, put, delete, list) to the **STT service IAM role**, while public access remains blocked

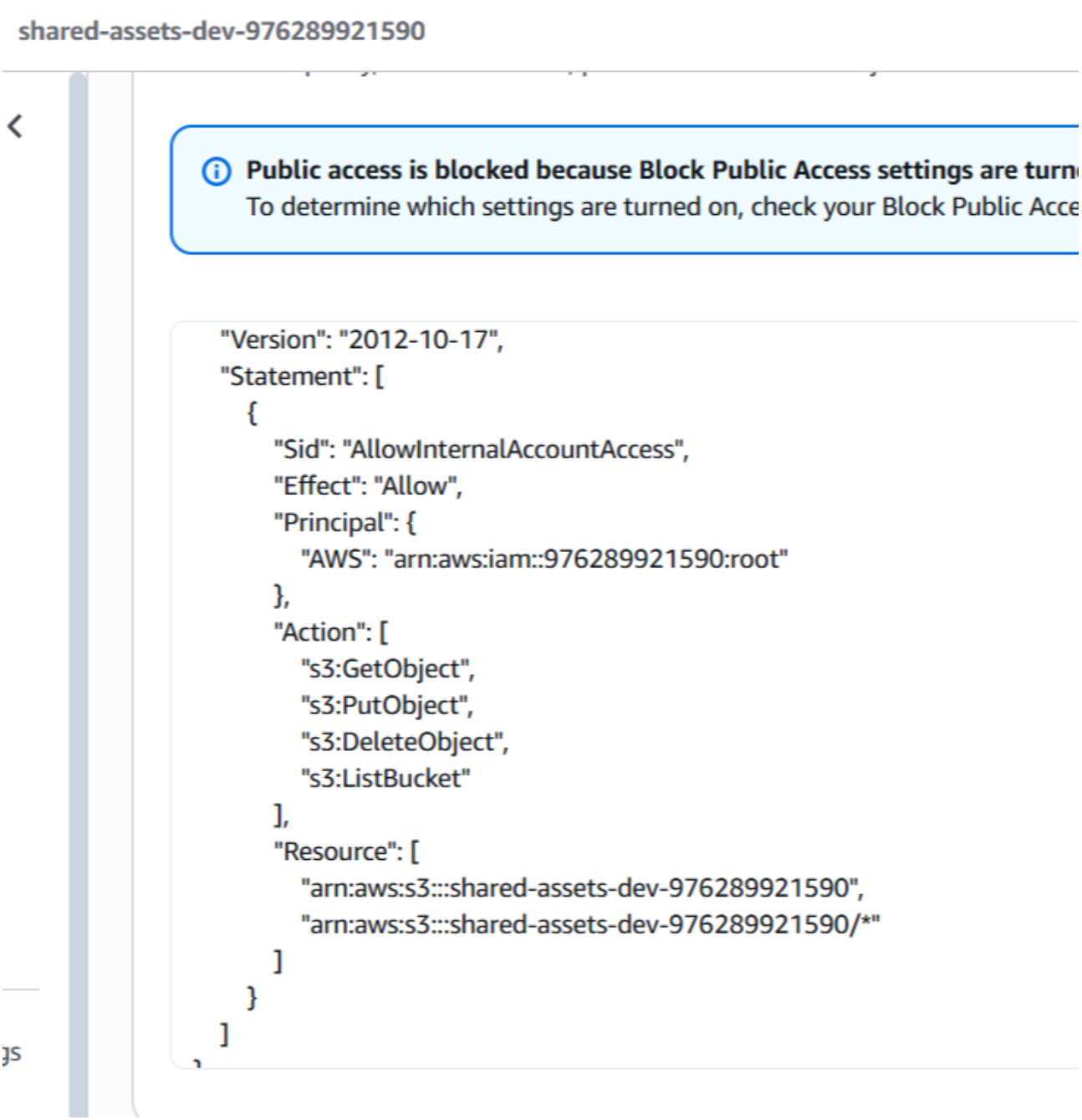


This policy grants full internal access (get, put, delete, list) to the root account for the TTS service bucket, while public access remains blocked

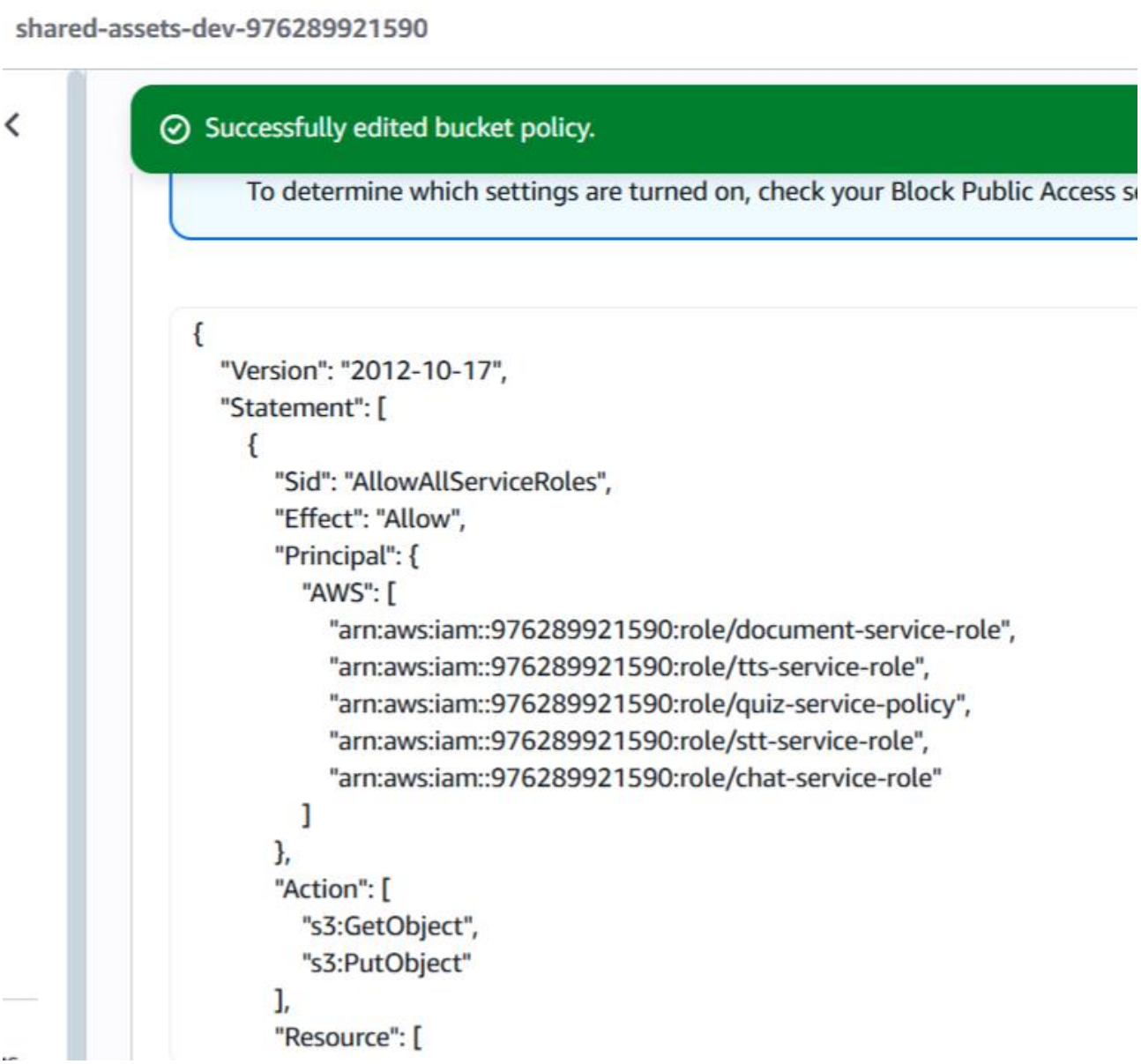
tts-service-storage-dev-976289921590



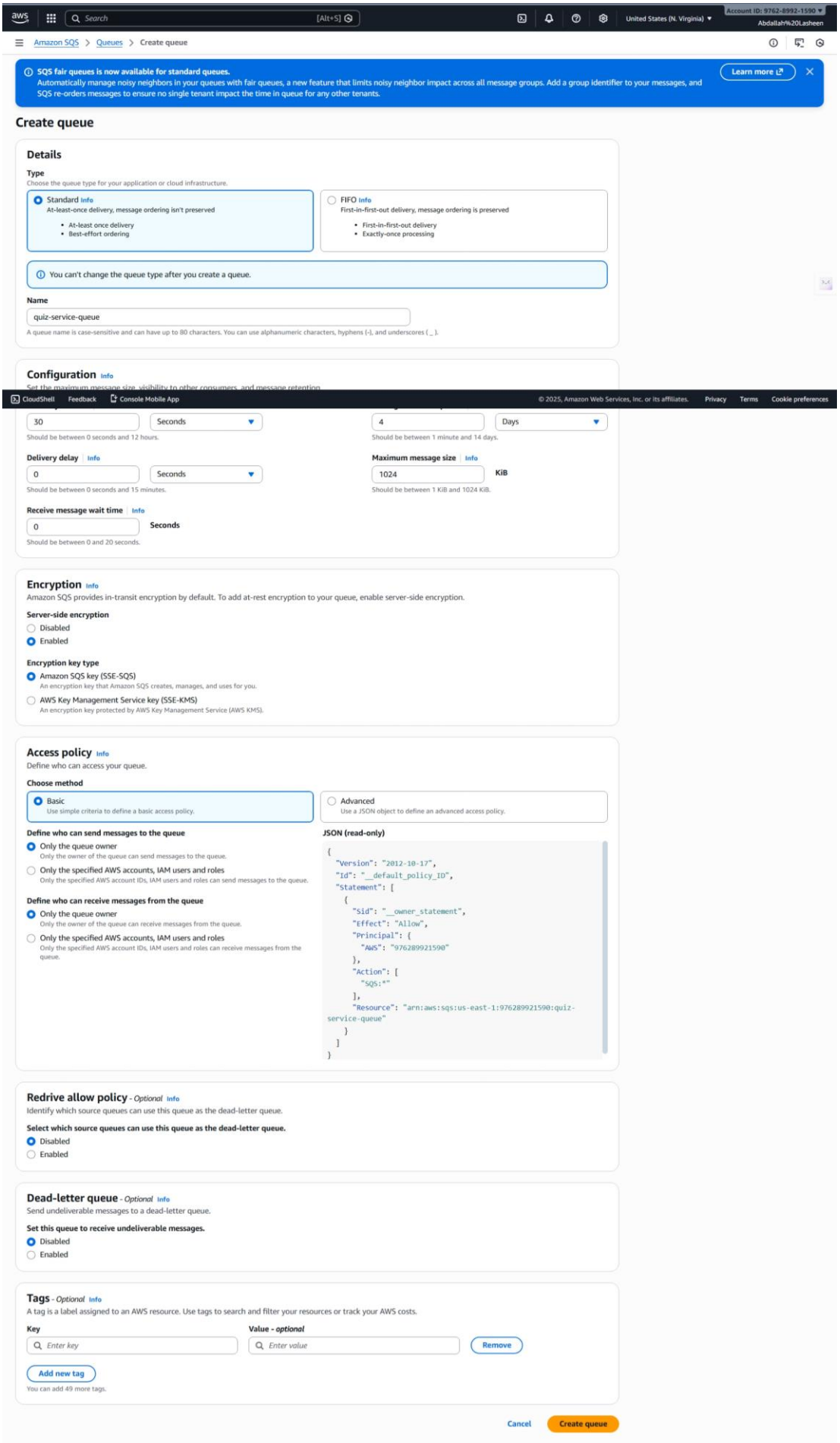
The bucket policy now grants full access (get, put, delete, list) to the **TTS service IAM role**, while public access remains blocked



This policy grants full internal access (get, put, delete, list) to the root account for the shared-assets bucket, while public access remains blocked

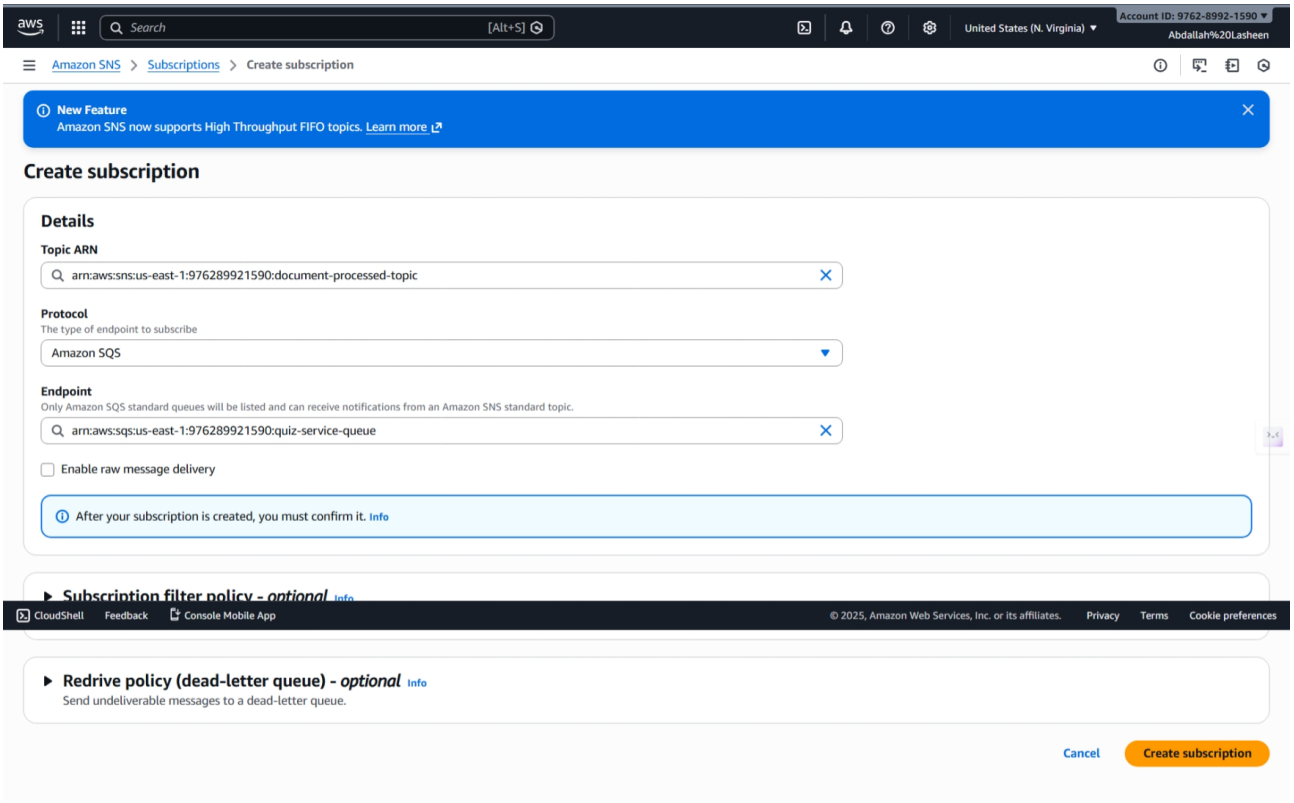


The shared-assets bucket now allows **read and write access** to all service-specific IAM roles (document, TTS, quiz, STT, chat), while public access remains blocked

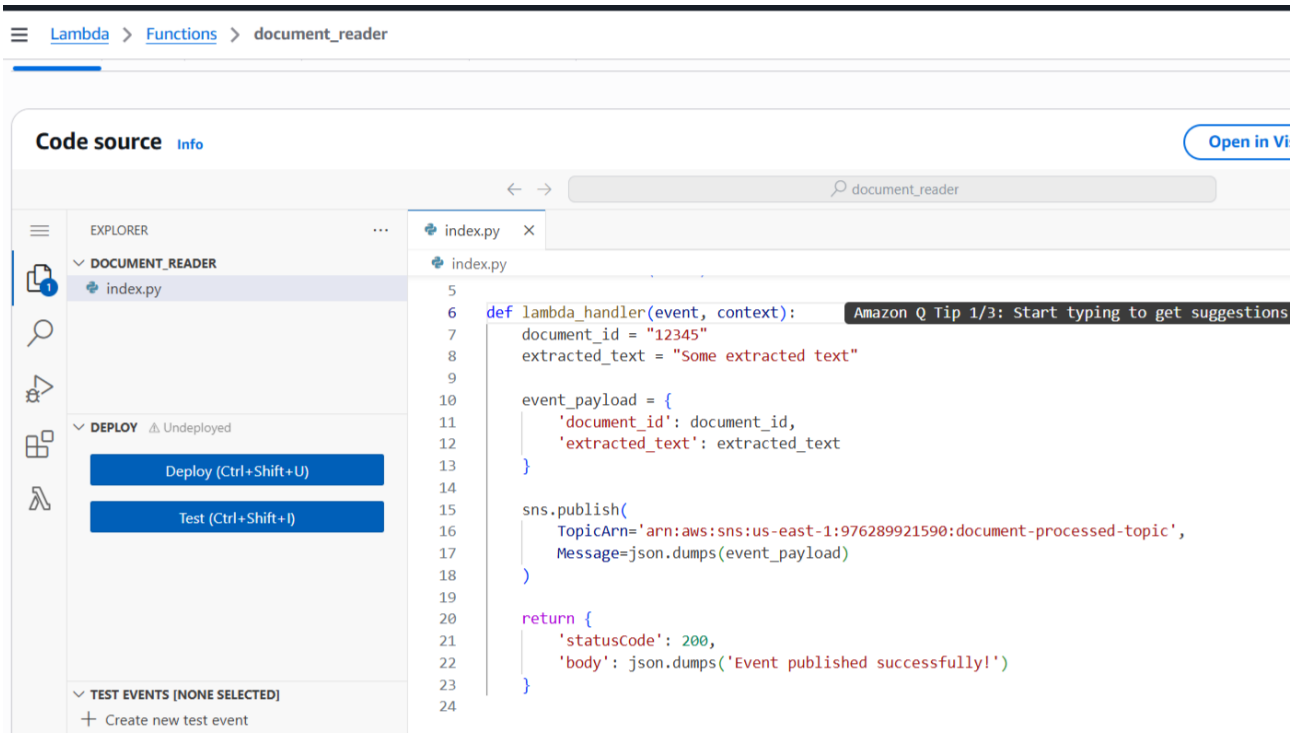

aws
Search
[Alt+S]
United States (N. Virginia)
Account ID: 9762-8992-1590
Abdallah%20Lasheen
Amazon SQS > Queues > Create queue
SQS fair queues is now available for standard queues.
Automatically manage noisy neighbors in your queues with fair queues, a new feature that limits noisy neighbor impact across all message groups. Add a group identifier to your messages, and SQS re-orders messages to ensure no single tenant impact the time in queue for any other tenants.
Learn more
Create queue
Details
Type
Choose the queue type for your application or cloud infrastructure.
Standard Info
At-least-once delivery, message ordering isn't preserved
At-least once delivery
Best-effort ordering
FIFO Info
First-in-first-out delivery, message ordering is preserved
First-in-first-out delivery
Exactly-once processing
You can't change the queue type after you create a queue.
Name
quiz-service-queue
A queue name is case-sensitive and can have up to 80 characters. You can use alphanumeric characters, hyphens (-), and underscores (_).
Configuration Info
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
30
Seconds
Should be between 0 seconds and 12 hours.
4
Days
Should be between 1 minute and 14 days.
Delivery delay Info
0
Seconds
Should be between 0 seconds and 15 minutes.
Maximum message size Info
1024
KiB
Should be between 1 KiB and 1024 KiB.
Receive message wait time Info
0
Seconds
Should be between 0 and 20 seconds.
Encryption Info
Amazon SQS provides in-transit encryption by default. To add at-rest encryption to your queue, enable server-side encryption.
Server-side encryption
Disabled
Enabled
Encryption key type
Amazon SQS key (SSE-SQS)
An encryption key that Amazon SQS creates, manages, and uses for you.
AWS Key Management Service key (SSE-KMS)
An encryption key protected by AWS Key Management Service (AWS KMS).
Access policy Info
Define who can access your queue.
Choose method
Basic
Use simple criteria to define a basic access policy.
Advanced
Use a JSON object to define an advanced access policy.
Define who can send messages to the queue
Only the queue owner
Only the owner of the queue can send messages to the queue.
Only the specified AWS accounts, IAM users and roles
Only the specified AWS account IDs, IAM users and roles can send messages to the queue.
Define who can receive messages from the queue
Only the queue owner
Only the owner of the queue can receive messages from the queue.
Only the specified AWS accounts, IAM users and roles
Only the specified AWS account IDs, IAM users and roles can receive messages from the queue.
JSON (read-only)
{
"Version": "2012-10-17",
"Id": "__default_policy_ID",
"Statement": [
{
"Sid": "__owner_statement",
"Effect": "Allow",
"Principal": {
"AWS": "976289921590"
},
"Action": [
"SQS:*"
],
"Resource": "arn:aws:sqs:us-east-1:976289921590:quiz-service-queue"
}
]
}
Redrive allow policy - Optional Info
Identify which source queues can use this queue as the dead-letter queue.
Select which source queues can use this queue as the dead-letter queue.
Disabled
Enabled
Dead-letter queue - Optional Info
Send undeliverable messages to a dead-letter queue.
Set this queue to receive undeliverable messages.
Disabled
Enabled
Tags - Optional Info
A tag is a label assigned to an AWS resource. Use tags to search and filter your resources or track your AWS costs.
Key
Value - optional
Enter key
Enter value
Remove
Add new tag
You can add 49 more tags.
Cancel
Create queue

This configuration creates a **Standard SQS queue named quiz-service-queue** with:

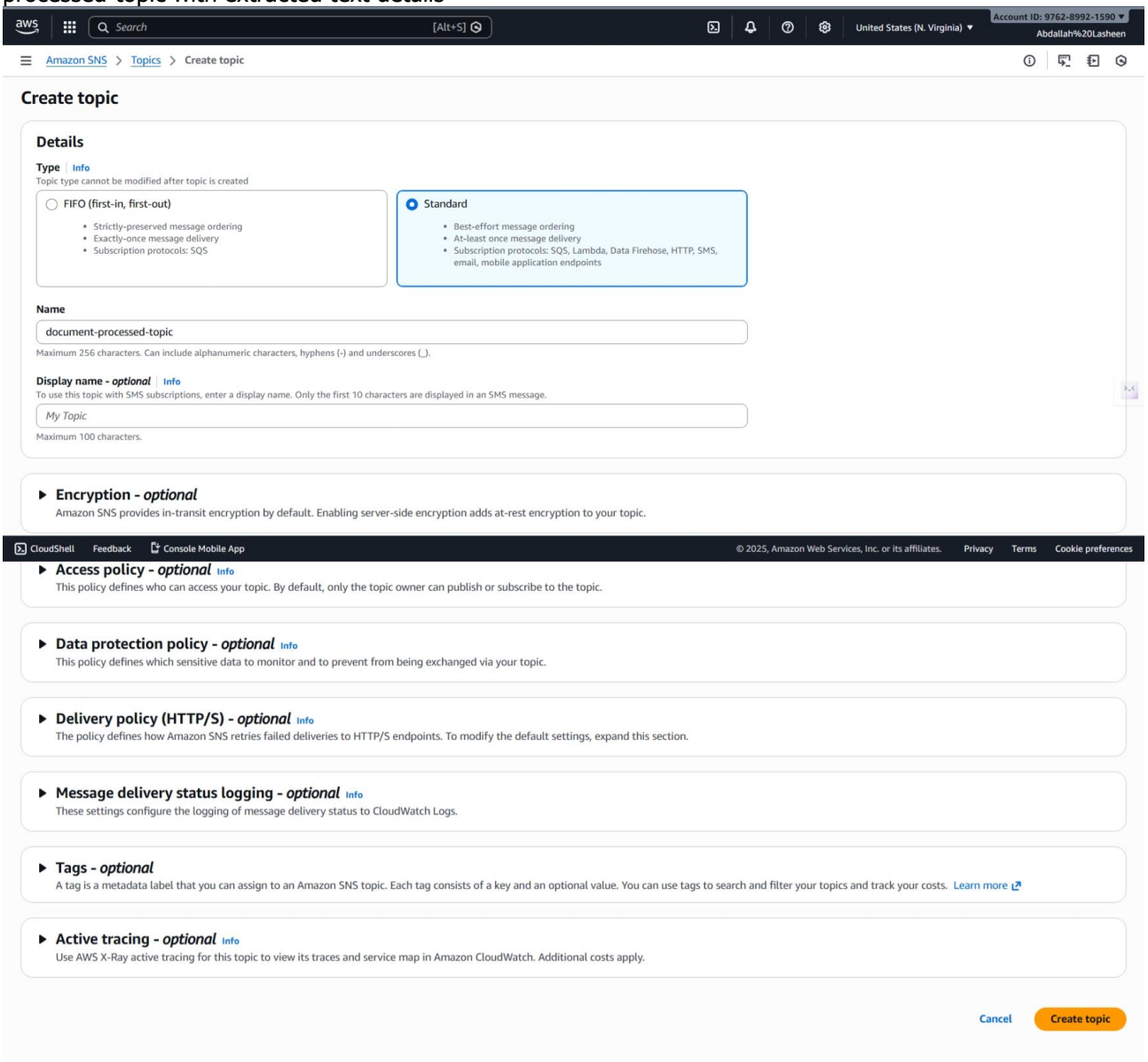
- No encryption,
- 4-day message retention,
- Open access policy (anyone can send/receive messages),
- No dead-letter queue,
- Tag: Owner = Merna.



This configuration creates an SNS subscription that sends messages from the **document-processed-topic** SNS topic to the **quiz-service-queue** SQS queue using the **Amazon SQS protocol**



This Lambda function simulates document processing and sends an SNS notification to the document-processed-topic with extracted text details



This configuration creates a **Standard SNS topic named document-processed-topic** with default settings (no encryption, no custom access policy, no tags).

AWS Data Sharing Mechanism Implementation Guide

---

1. Introduction

In this project, services are designed to have **completely isolated storage** and **share data only via events**.
Main services:

- **Document Reader Service**
- **Quiz Service**

---

2. System Architecture

[User]
↓ Upload
[Document Reader Lambda]
↓ Store document in S3 bucket
↓ Publish event to SNS
[Quiz Service Lambda]

↓ Consume from SQS
↓ Store relevant data in its own DB/S3

---

3. Document Reader Service

3.1 S3 Bucket

- Each service has a **dedicated bucket**.
- Example: document-reader-storage-dev-976289921590
- Stores uploaded documents.

3.2 Lambda Function

- Runtime: **Python 3.11**
- Code:

```
import boto3
import json

sns = boto3.client("sns")
s3 = boto3.client("s3")

TOPIC_ARN = "arn:aws:sns:us-east-1:976289921590:document-processed-topic"
BUCKET_NAME = "document-reader-storage-dev-976289921590"

def lambda_handler(event, context):
    # Store document in S3
    document_id = "12345"
    extracted_text = "Some extracted text"
    document_content = b"Sample file content"

    s3.put_object(
        Bucket=BUCKET_NAME,
        Key=f"{document_id}.txt",
        Body=document_content
    )

    # Publish event to SNS
    payload = {
        "document_id": document_id,
        "extracted_text": extracted_text
    }

    sns.publish(
        TopicArn=TOPIC_ARN,
        Message=json.dumps(payload)
    )

    return {
        "statusCode": 200,
        "body": "Event published to SNS and file stored in S3"
    }
```

3.3 Permissions

- Lambda Execution Role must include:
    - **AmazonS3FullAccess** (or limited to bucket)
    - **AmazonSNSFullAccess**
    - **AWSLambdaBasicExecutionRole**

---

4. SNS Topic

- Type: **Standard**
- Name: document-processed-topic
- Purpose: Publish events when a document is processed.

5. Quiz Service

5.1 SQS Queue

- Receives events from SNS:
  - Queue Name: quiz-service-queue
  - Subscribed to SNS Topic: document-processed-topic
  - Each service has its own dedicated queue.

5.2 Lambda Function

- Handles events from SQS
- Stores relevant data in **S3 or service-specific database**

Example Code:

```
import boto3
import json

sqs = boto3.client("sqs")
s3 = boto3.client("s3")

QUEUE_URL = "https://sqs.us-east-1.amazonaws.com/976289921590/quiz-service-queue"
BUCKET_NAME = "quiz-service-storage-dev-976289921590"

def lambda_handler(event, context):
  for record in event['Records']:
    message = json.loads(record['body'])
    document_id = message['document_id']
    extracted_text = message['extracted_text']

    # Store in service-specific S3
    s3.put_object(
      Bucket=BUCKET_NAME,
      Key=f"{document_id}.txt",
      Body=extracted_text.encode('utf-8')
    )

  return {
    'statusCode': 200,
    'body': 'Data processed and stored successfully'
  }
```

5.3 Permissions

- Lambda Execution Role must include:
  - **AmazonS3FullAccess** (or limited to bucket)
  - **AmazonSQSFullAccess**
  - **AWSLambdaBasicExecutionRole**

---

6. Key Principles

1. **Isolated storage**: Each service has its own S3 bucket and database.
2. **Event-driven data sharing**: No direct database queries or S3 access between services.
3. **Decoupling**: Services communicate only via SNS/SQS events.

Infrastructure as Code (IaC) is an approach used to manage and provision cloud infrastructure through machine-readable configuration files instead of manual processes. On AWS, this approach ensures consistency, scalability, and automation across different environments.

2. Tool Selection: AWS CloudFormation

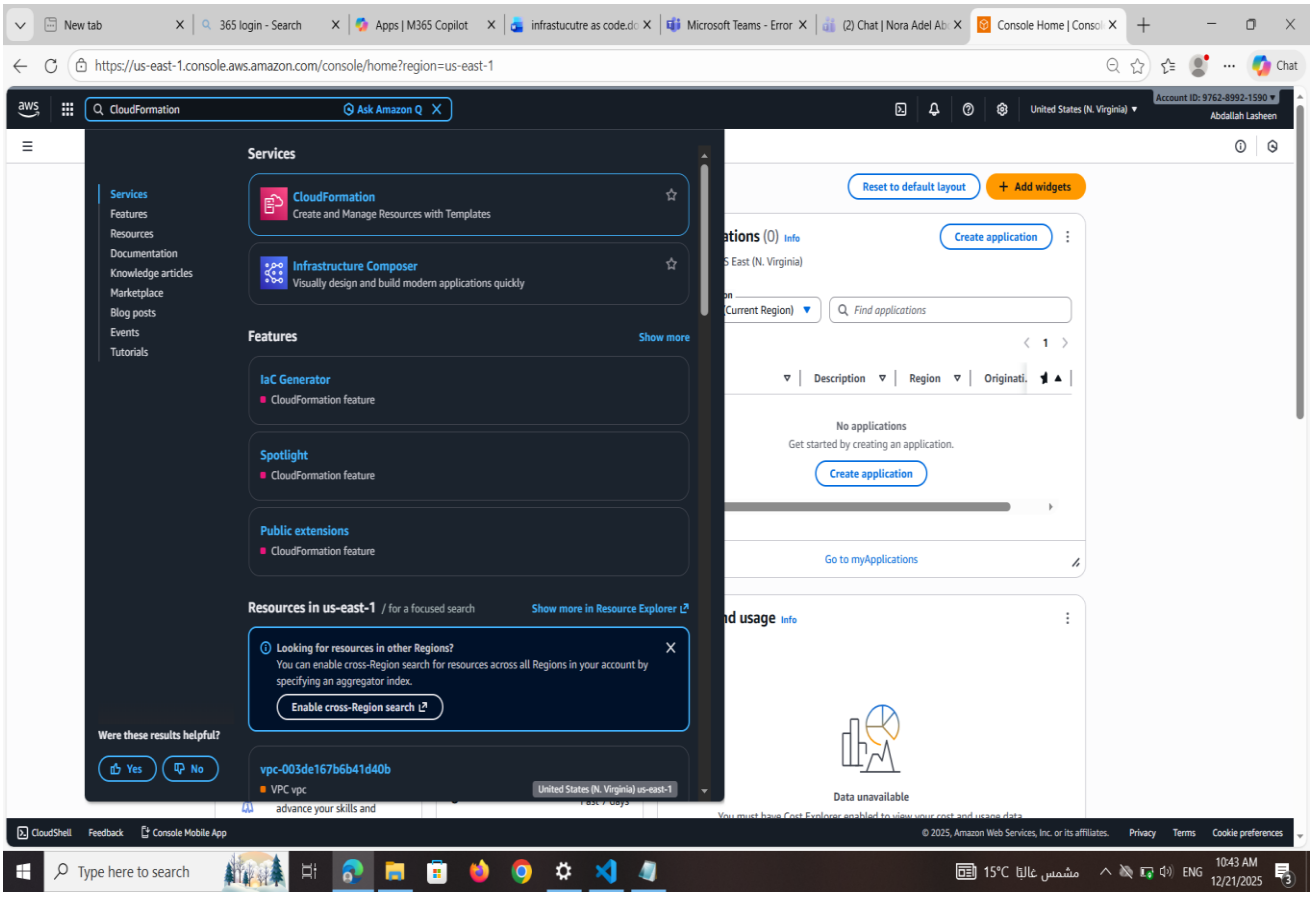
AWS CloudFormation is used as the Infrastructure as Code tool because it is a native AWS service that allows infrastructure to be defined using YAML or JSON templates. CloudFormation manages the creation, update, and deletion of AWS resources in a controlled and automated manner.

3. Step 1: Access AWS CloudFormation

- Log in to the AWS Management Console.
- Select the required AWS Region.
- Search for CloudFormation from the AWS services menu.
- Open the CloudFormation dashboard.

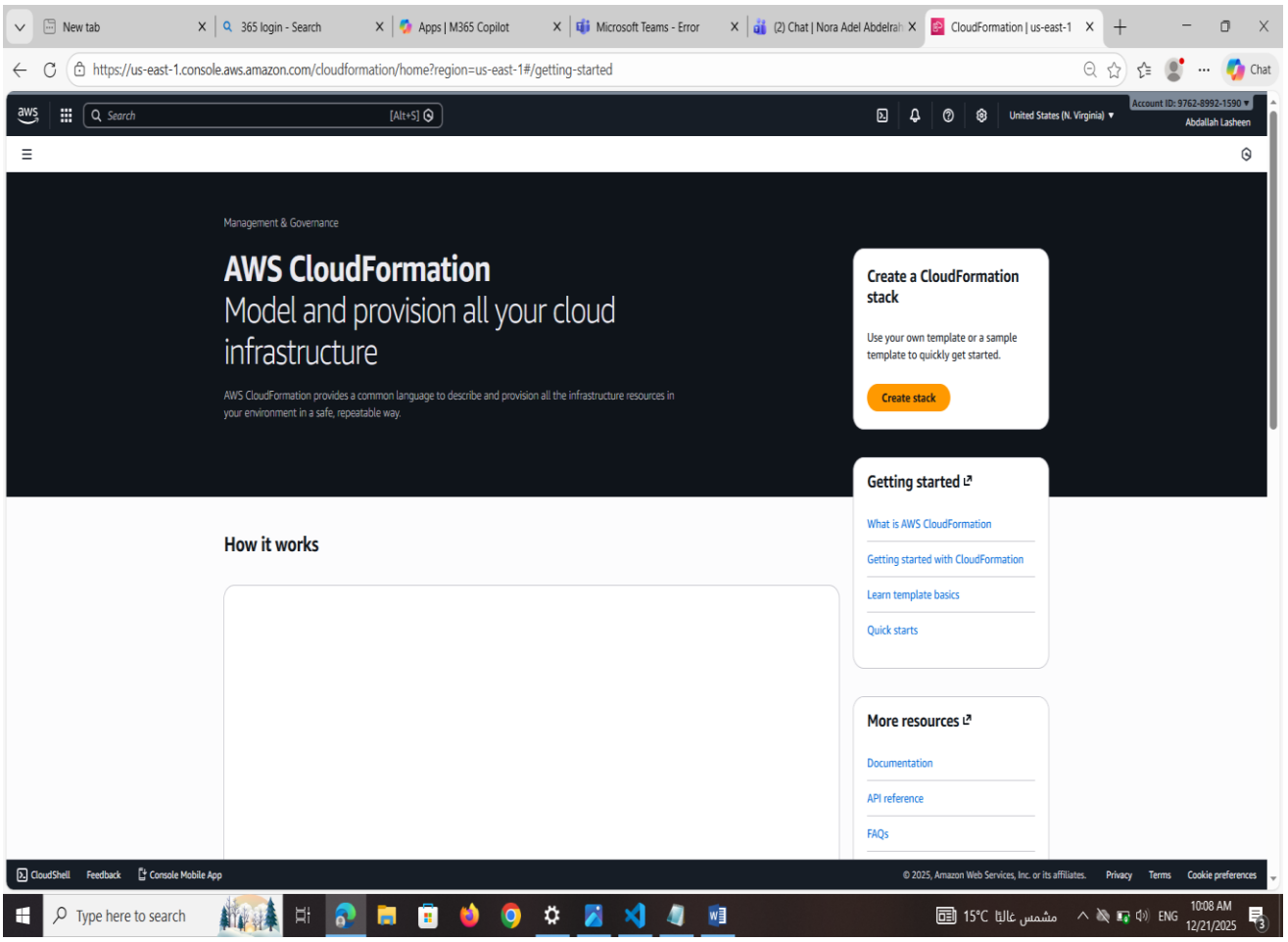
4. Step 2: Define Infrastructure Using Templates

Infrastructure is defined using CloudFormation templates written in YAML. Each template represents a specific infrastructure component.
Typical components include:


https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1
CloudFormation
Ask Amazon Q
United States (N. Virginia)
Abdallah Lasheen
Services
Features
Resources
Documentation
Knowledge articles
Marketplace
Blog posts
Events
Tutorials
CloudFormation
Create and Manage Resources with Templates
Infrastructure Composer
Visually design and build modern applications quickly
Features
Show more
IaC Generator
CloudFormation feature
Spotlight
CloudFormation feature
Public extensions
CloudFormation feature
Resources in us-east-1 / for a focused search
Show more in Resource Explorer
Looking for resources in other Regions?
You can enable cross-Region search for resources across all Regions in your account by specifying an aggregator index.
Enable cross-Region search
Were these results helpful?
Yes
No
vpc-003de167b6b41d40b
VPC vpc
Reset to default layout
Add widgets
Create application
Find applications
Description
Region
No applications
Get started by creating an application.
Create application
Go to myApplications
Data unavailable
CloudShell
Feedback
Console Mobile App
Privacy
Terms
Cookie preferences
Type here to search
15°C
ENG
10:43 AM
12/21/2025

- Networking (VPC, Subnets)
- Compute (EC2)
- Security (Security Groups, IAM roles)

This code-based definition ensures repeatable and version-controlled infrastructure.



5. Step 3: Implement Modular Infrastructure Components

To implement modular infrastructure:

- Each infrastructure component is defined in a separate template.

Examples:

- VPC template
- EC2 template
- Database template

Each module can be reused independently, making the infrastructure easier to manage and extend.

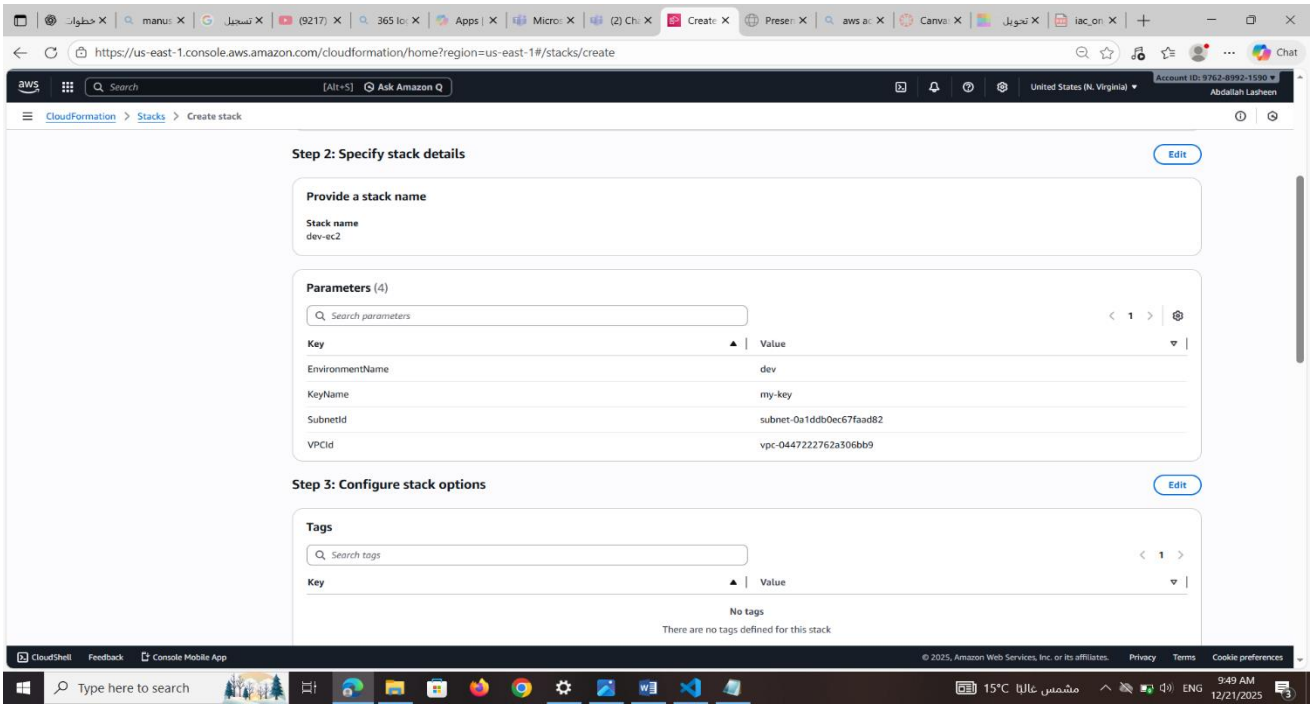
6. Step 4: Create a Stack for Each Module

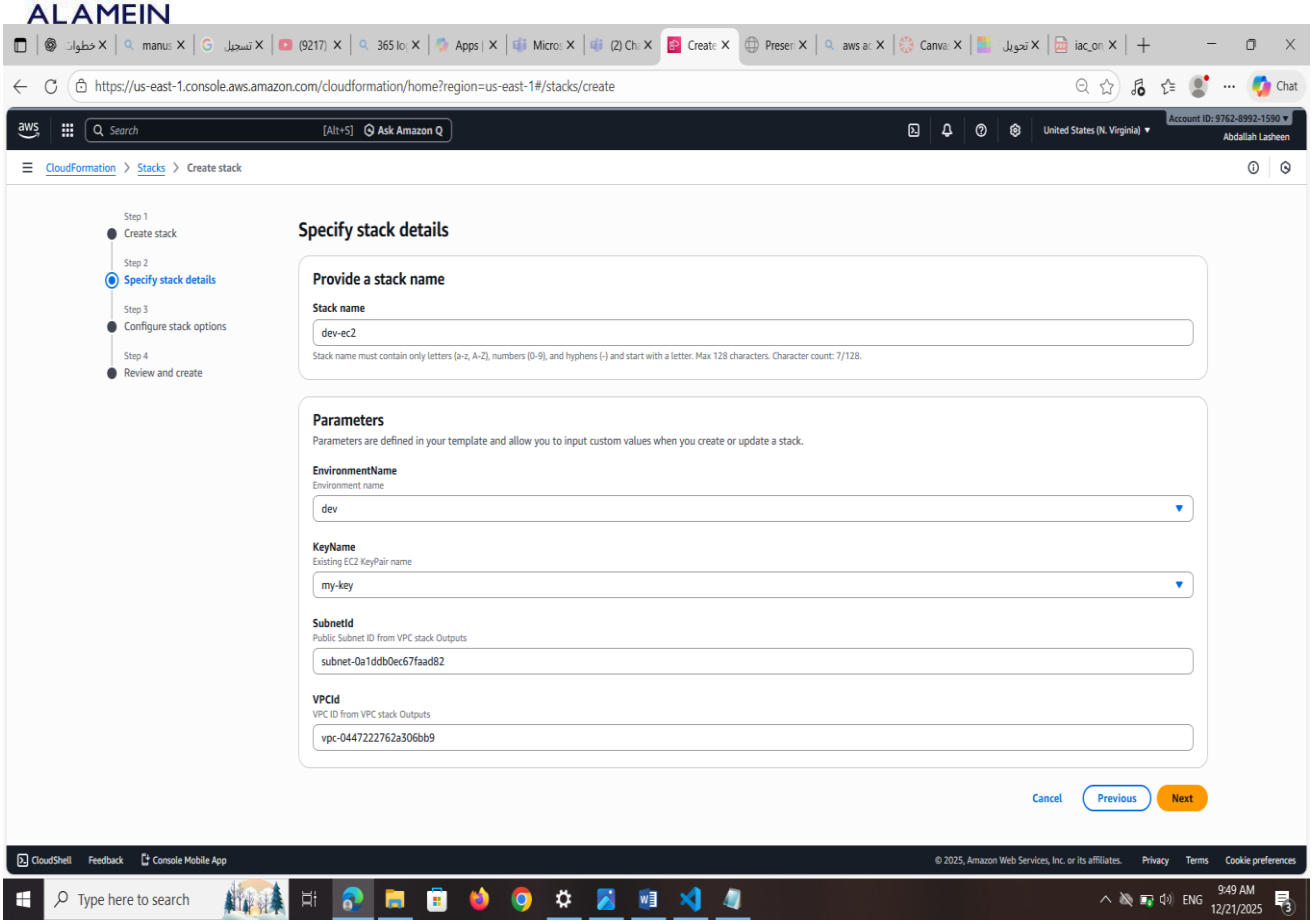
In CloudFormation, each template is deployed as a stack. Steps:

- Click Create stack.
- Upload the module template (for example, the VPC template).
- Provide required parameters.
- Create the stack.

This modular approach improves flexibility and maintenance

.



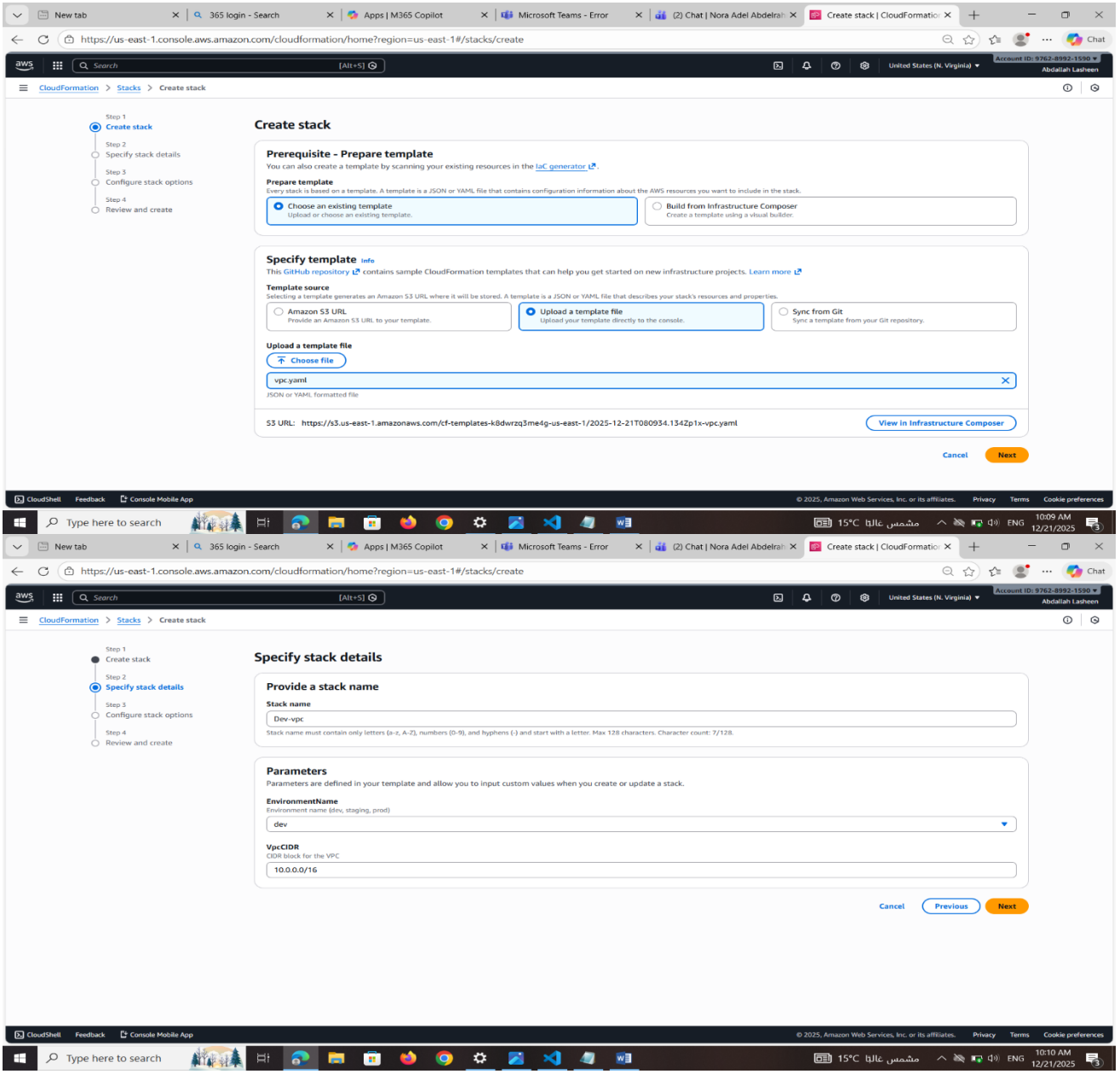

https://us-east-1.console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks/create
CloudFormation > Stacks > Create stack
Step 1
Create stack
Step 2
Specify stack details
Step 3
Configure stack options
Step 4
Review and create
Specify stack details
Provide a stack name
Stack name
dev-ec2
Stack name must contain only letters (a-z, A-Z), numbers (0-9), and hyphens (-) and start with a letter. Max 128 characters. Character count: 7/128.
Parameters
Parameters are defined in your template and allow you to input custom values when you create or update a stack.
EnvironmentName
Environment name
dev
KeyName
Existing EC2 KeyPair name
my-key
SubnetId
Public Subnet ID from VPC stack Outputs
subnet-0a1ddb0ec67faad82
VPCId
VPC ID from VPC stack Outputs
vpc-0447222762a306bb9
Cancel
Previous
Next
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Type here to search
ENG
9:49 AM
12/21/2025

7. Step 5: Separate Environments Using Parameters

Environment separation is achieved by using parameters in the templates. Example environments:

- Development (dev)
- Staging
- Production (prod) Each environment uses:
- The same template
- Different parameter values
- A separate CloudFormation stack

This ensures complete isolation between environments.

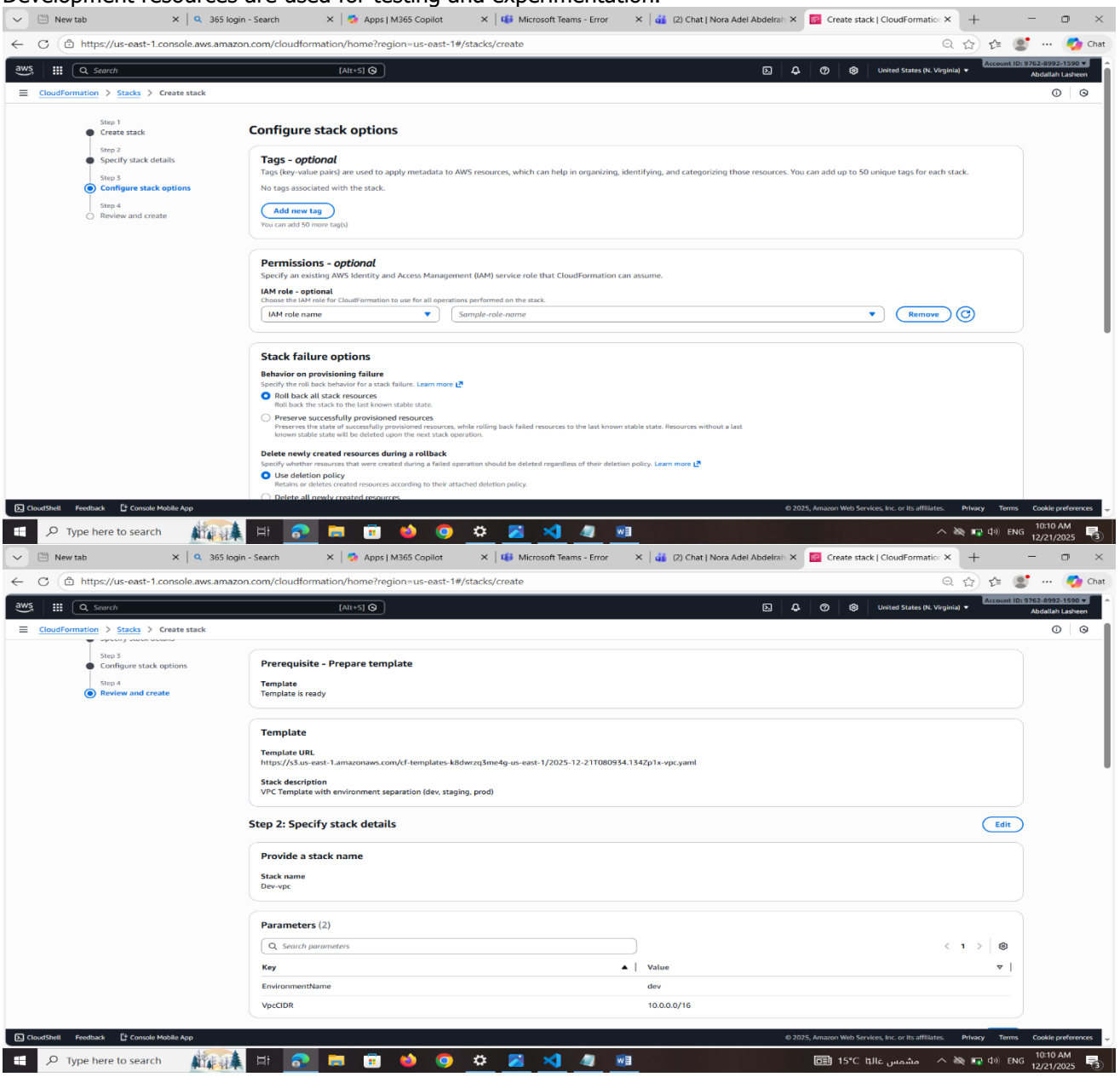


8. Step 6: Deploy the Development Environment

For the development environment:

- Create a stack named dev-vpc.
- Set the parameter EnvironmentName to dev.
- Deploy the stack.

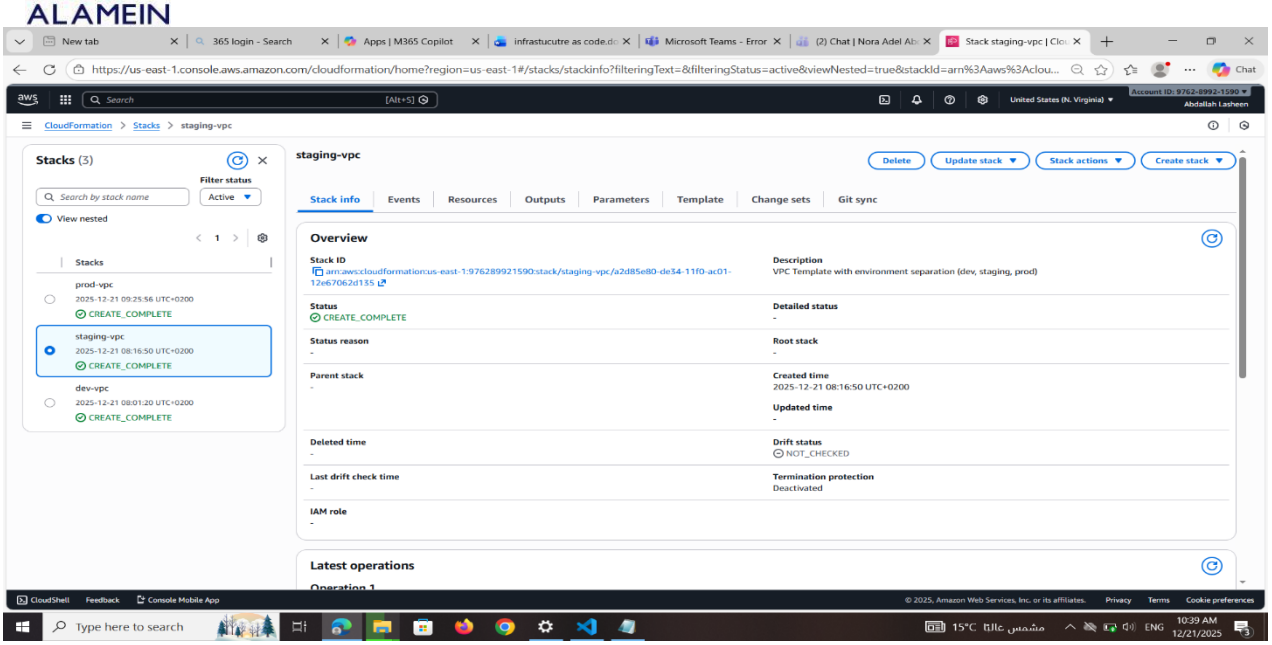
Development resources are used for testing and experimentation.



9. Step 7: Deploy the Staging Environment

For the staging environment:

- Create a stack named staging-vpc.
- Set the parameter EnvironmentName to staging.
- Deploy the stack.


Stacks (3)
staging-vpc
Delete
Update stack
Stack actions
Create stack
Stack info
Events
Resources
Outputs
Parameters
Template
Change sets
Git sync
Overview
Latest operations

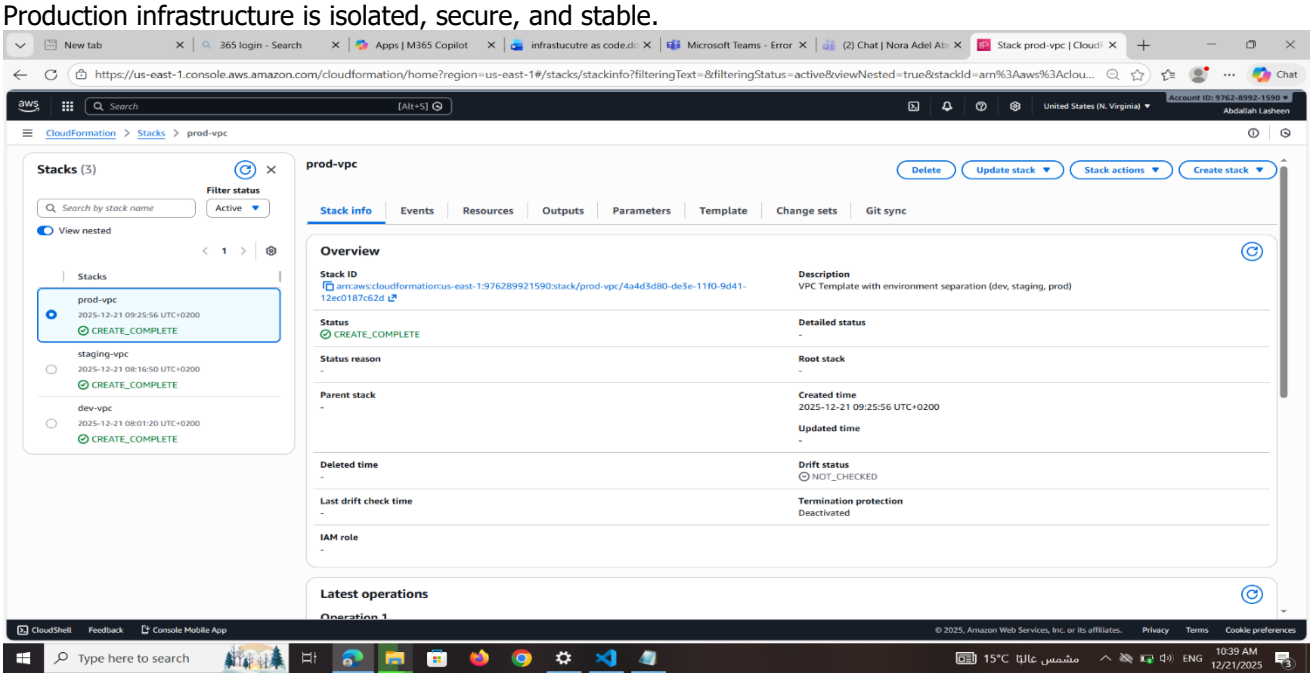
Staging closely matches production and is used for final validation.

10. Step 8: Deploy the Production Environment

For the production environment:

- Create a stack named prod-vpc.
- Set the parameter EnvironmentName to prod.
- Deploy the stack.

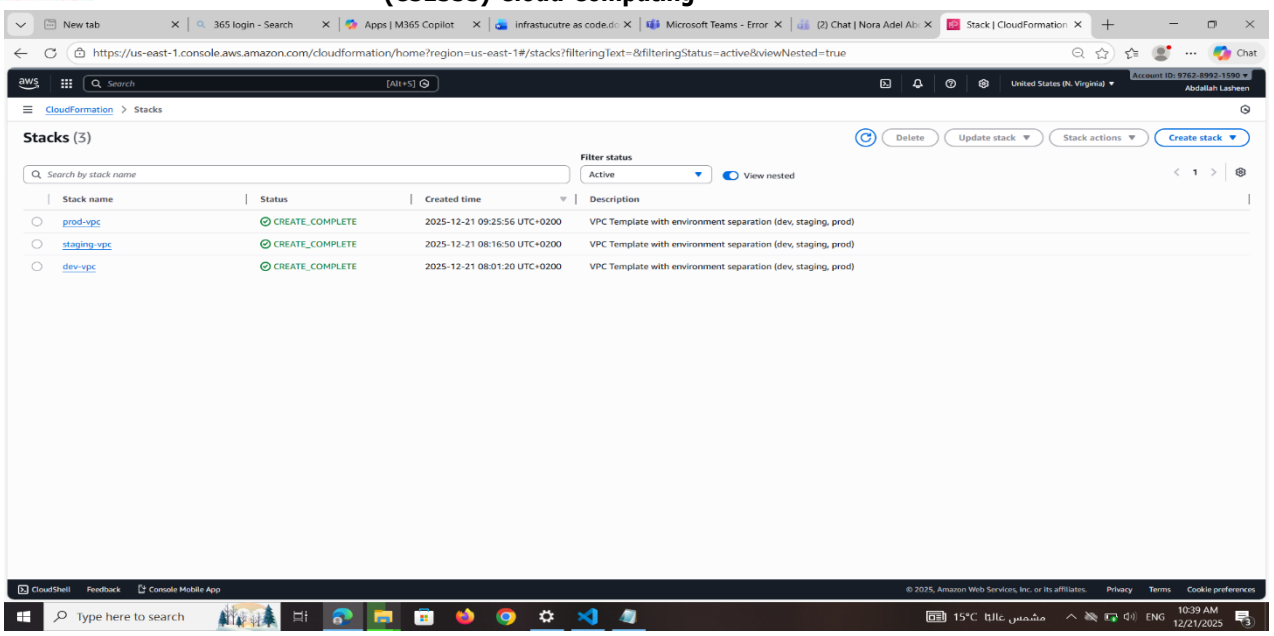
Production infrastructure is isolated, secure, and stable.



11. Step 9: Review and Create the Stack

Before deployment:

- Review all stack details.
- Acknowledge required AWS permissions.
- Create the stack.



12. Step 10:code

```
AWSTemplateFormatVersion: '2010-09-09'
Description: EC2 Instance using existing VPC and Subnet

Parameters:
  EnvironmentName:
    Type: String
    AllowedValues:
      - dev
      - staging
      - prod
    Description: Environment name

  VPCId:
    Type: String
    Description: VPC ID from VPC stack Outputs

  SubnetId:
    Type: String
    Description: Public Subnet ID from VPC stack Outputs

  KeyName:
    Type: AWS::EC2::KeyPair::KeyName
    Description: Existing EC2 KeyPair name

Resources:
  InstanceSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Allow SSH access
      VpcId: !Ref VPCId
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0
      Tags:
        - Key: Name
          Value: !Sub "${EnvironmentName}-sg"
```

```
  MyInstance:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: t2.micro
      ImageId: ami-0c02fb55956c7d316   # Amazon Linux 2 (us-east-1)
      KeyName: !Ref KeyName
      SubnetId: !Ref SubnetId
      SecurityGroupIds:
        - !Ref InstanceSecurityGroup
      Tags:
        - Key: Name
          Value: !Sub "${EnvironmentName}-ec2"

Outputs:
  InstanceId:
    Description: EC2 Instance ID
    Value: !Ref MyInstance
```

1. Network Security

1.1 Private Subnets

- All services are deployed in **private subnets**.
- Direct internet access is restricted.

1.2 Internet Exposure

- Only the **Application Load Balancer (ALB)** is exposed to the internet.
- All other resources remain in private subnets.

1.3 Security Groups

- Implement **least privilege** rules.
- Only allow required inbound/outbound traffic.

1.4 Monitoring

- Enable **VPC Flow Logs** to monitor traffic.

1.5 Web Protection

- Enable **AWS WAF** on ALB to block malicious requests.
- Implement **DDoS protection** with **AWS Shield**.

---

2. Data Security

2.1 Encryption at Rest

- Encrypt all data in **S3, EBS, and RDS**.
- Use **AWS KMS** for encryption key management.
- Maintain **separate keys per service**.
- Implement **regular key rotation**.

2.2 Encryption in Transit

- Use **TLS 1.3** for all communication.

2.3 Secrets Management

- Store sensitive credentials in **AWS Secrets Manager**.
- Avoid hard-coding secrets in code or containers.

---

4. Access Control

Service	IAM Role	Trusted Entity	AWS Service	Allowed Actions

Chat Service	chat-service-role	ecs-tasks.amazonaws.com	S3	GetObject, PutObject, DeleteObject, ListBucket
Chat Service	chat-service-role	ecs-tasks.amazonaws.com	CloudWatch Logs	CreateLogGroup, CreateLogStream, PutLogEvents
Lambda Service	lambda-exec-role	lambda.amazonaws.com	S3	GetObject, PutObject
Lambda Service	lambda-exec-role	lambda.amazonaws.com	CloudWatch Logs	CreateLogGroup, CreateLogStream, PutLogEvents
EC2 Service	ec2_platform_role	ec2.amazonaws.com	RDS	DescribeDBInstances, Connect

Service	IAM Role	Trusted Entity	AWS Service	Allowed Act
Document Reader	document_reader-role-gpgftvdt	lambda.amazonaws.com	SNS	sns:*

Document Reader	document_reader-role-gpgftvdt	lambda.amazonaws.com	SMS via SNS	sms-voice:S voice:Verify
Document Reader	document_reader-role-gpgftvdt	lambda.amazonaws.com	CloudWatch Logs	CreateLogG PutLogEven
Service	IAM Role	Trusted Entity	AWS Service	Allowed Act
TTS Service	tts-service-role	ecs-tasks.amazonaws.com	S3	GetObject, I ListBucket
TTS Service	tts-service-role	ecs-tasks.amazonaws.com	CloudWatch Logs	CreateLogG PutLogEven

TTS Service	tts-service-role	ecs-tasks.amazonaws.com	Polly	SynthesizeS

Service	IAM Role	Trusted Entity	AWS Service	Allowed Act
Quiz Service	quiz-service-role	ecs-tasks.amazonaws.com	S3	GetObject, I ListBucket
Quiz Service	quiz-service-role	ecs-tasks.amazonaws.com	Secrets Manager	GetSecretVa

Service	IAM Role	Trusted Entity	AWS Service	Allowed Act
Container Service	container_service_account_role	ecs-tasks.amazonaws.com	Elastic Container Registry	GetAuthoriz BatchCheck GetDownloa

Service	IAM Role	Trusted Entity	AWS Service	Allowed Act
Document Reader	document_reader-role-gpgftvdt	lambda.amazonaws.com	SNS	* (All SNS a voice:Descri CreateVerifi SendTextMe DeleteVerifi VerifyDestin DescribeAcc DescribeSpe DescribePho SetTextMes DescribeOpt DeleteOpted
Document Reader	document_reader-role-gpgftvdt	lambda.amazonaws.com	CloudWatch Logs	CreateLogG PutLogEven

Service	IAM Role	Trusted Entity	AWS Service	Allowed Act
Lambda Service	lambda-exec-role	lambda.amazonaws.com	SNS	* (Full acces

Lambda Service	lambda-exec-role	lambda.amazonaws.com	End User Messaging SMS	DescribeVer CreateVerifi SendDestina SendTextMe DeleteVerifi VerifyDestin DescribeAcc DescribeSpe DescribePho SetTextMes DescribeOpt DeleteOpte
Service	IAM Role	Trusted Entity	AWS Service	Allowed Act
EC2 Service	terraform-ec2-role	ec2.amazonaws.com	EC2	*
EC2 Service	terraform-ec2-role	ec2.amazonaws.com	S3	Get*, List*,
EC2 Service	terraform-ec2-role	ec2.amazonaws.com	CloudWatch	*

EC2 Service	terraform-ec2-role	ec2.amazonaws.com	Auto Scaling	*
EC2 Service	terraform-ec2-role	ec2.amazonaws.com	ELB / ELB v2	*
EC2 Service	terraform-ec2-role	ec2.amazonaws.com	IAM	CreateServi