Configuration Manual

MSc Research Project
Cloud Computing

Deepak Venkatesh Babu
Student ID: X23267101

School of Computing
National College of Ireland

Supervisor: Prof. Aqeel Kazmi

National College of Ireland
MSc Project Submission Sheet
School of Computing

Student Name: Deepak Venkatesh Babu

Student ID: X23267101

Programme: MSCCLOUD **Year:** 2024

Module: MSCCLOUD Research Project

Lecturer: Prof. Aqeel Kazmi

Submission Due Date: 15/09/2025

Project Title: EZTSM: Enhanced Zero Trust Security Model with ML based anomaly detection

Word Count: 1308

Page Count: 9

Signature: Deepak Venkatesh Babu

Date: 15/09/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Deepak Venkatesh Babu
Student ID: X23267101

1 Requirements

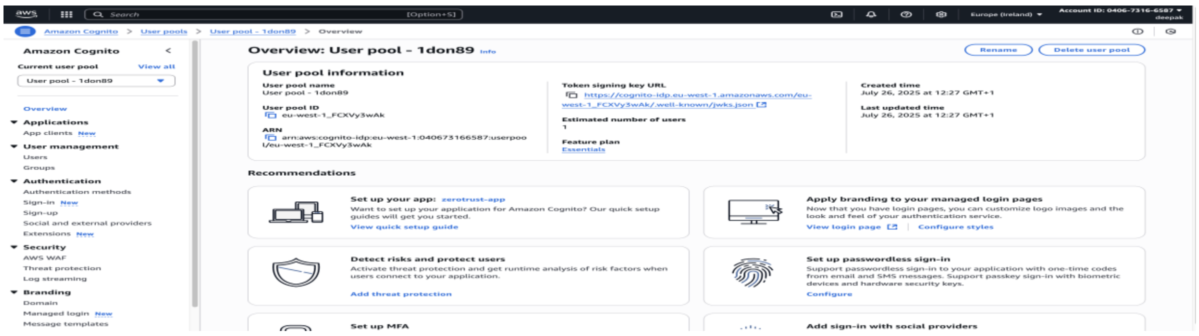
The implementation of Enhanced Zero Trust Security Model (EZTSM) required various tools and programs. This configuration manual shows the step by step process for setting up the environment and deployment to achieve the desired result. The setup is divided into two major categories (1) AWS environment setup (2) Anomaly Detection model training setup.

1.1 AWS environment setup

In this research, the standard AWS account and a user with required access has been created. The user has both programmatic and console access. All the AWS services were created through console only. Steps for creating required AWS services and integrating that between services are mentioned below.

Amazon Cognito

1. In the Cognito dashboard create user pool, select Single Page Application (SPA) under application type because this type won't create any client secret which we don't require.
2. Under configure option, for sign in attributes "Email" and "Username" has been provided and for signup attributes "email" and "name" were given.
3. This will create a User Pool and App Client shown in Figure 1 and 2.
4. An user is created with the required attributes, with email verified, shown in Figure 3.



Overview: User pool - 1don89 Info

User pool information

User pool name
User pool - 1don89

User pool ID
eu-west-1_FCXVy3wAk

ARN
arn:aws:cognito-idp:eu-west-1:040673166587:userpool/eu-west-1_FCXVy3wAk

Token signing key URL
https://cognito-idp.eu-west-1.amazonaws.com/eu-west-1_FCXVy3wAk/.well-known/jwks.json

Estimated number of users
1

Feature plan
Essentials

Created time
July 26, 2025 at 12:27 GMT+1

Last updated time
July 26, 2025 at 12:27 GMT+1

Recommendations

Set up your app: zerotrust-app
Want to set up your application for Amazon Cognito? Our quick setup guides will get you started.
View quick setup guide

Apply branding to your managed login pages
Now that you have login pages, you can customize logo images and the look and feel of your authentication service.
View login page | Configure styles

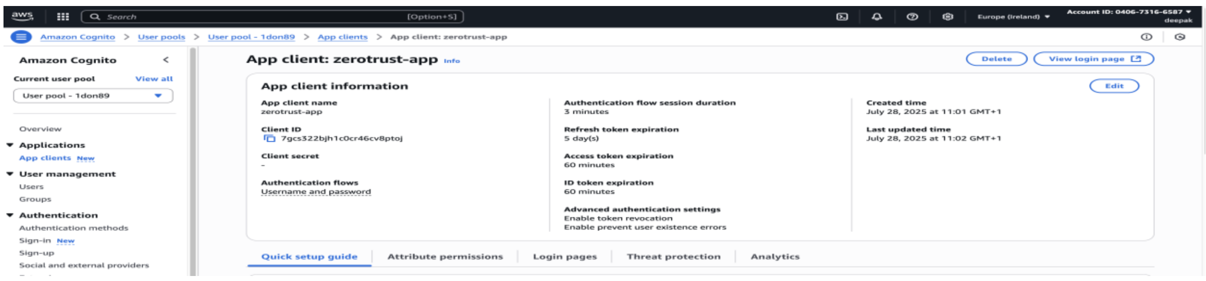
Detect risks and protect users
Activate threat protection and get runtime analysis of risk factors when users connect to your application.
Add threat protection

Set up passwordless sign-in
Support passwordless sign-in to your application with one-time codes from email and SMS messages. Support passkey sign-in with biometric devices and hardware security keys.
Configure

Set up MFA

Add sign-in with social providers

Figure 1: User pool dashboard



App client: zerotrust-app Info

App client information

App client name
zerotrust-app

Client ID
7gcs322bjh1c0cr46cv8ptoj

Client secret
-

Authentication flows
Username and password

Authentication flow session duration
3 minutes

Refresh token expiration
5 day(s)

Access token expiration
60 minutes

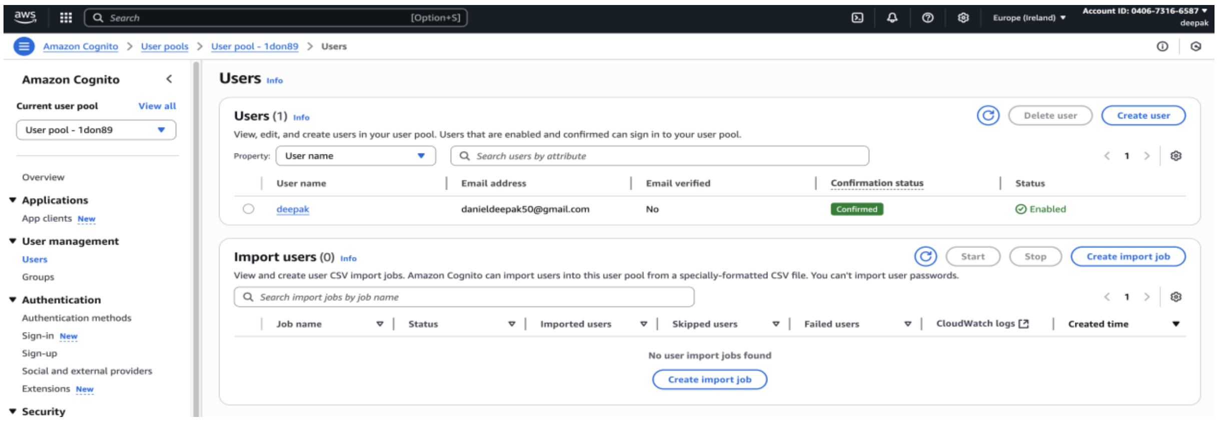
ID token expiration
60 minutes

Advanced authentication settings
Enable token revocation
Enable prevent user existence errors

Created time
July 28, 2025 at 11:01 GMT+1

Last updated time
July 28, 2025 at 11:02 GMT+1

Figure 2: App client page



Users Info

Users (1) Info

View, edit, and create users in your user pool. Users that are enabled and confirmed can sign in to your user pool.

Property: User name | Search users by attribute

User name	Email address	Email verified	Confirmation status	Status
deepak	danieldeepak50@gmail.com	No	Confirmed	✓ Enabled

Import users (0) Info

View and create user CSV import jobs. Amazon Cognito can import users into this user pool from a specially-formatted CSV file. You can't import user passwords.

Search import jobs by job name

Job name	Status	Imported users	Skipped users	Failed users	CloudWatch logs	Created time

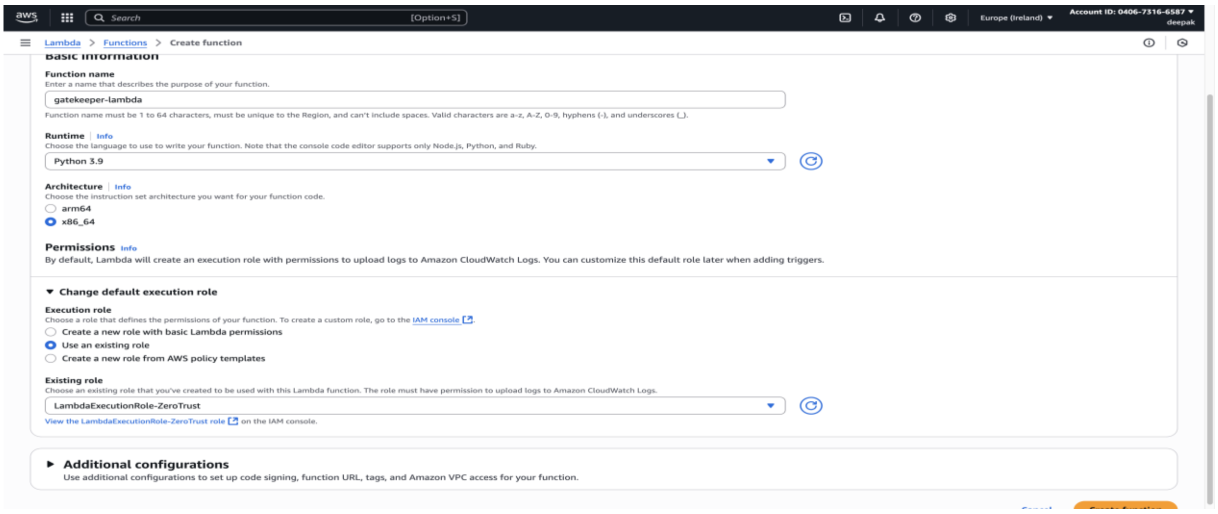
No user import jobs found

Create import job

Figure 3: Users dashboard

AWS Lambda

1. In this project two lambda functions are created, one act as Gatekeeper “gatekeeper-lambda” and another as business logic lambda “real_lambda”.
2. In the lambda creation page shown in figure 4 mention function name, runtime is selected as python 3.9, default execution role is “LambdaExecutionRole-ZeroTrust” which has necessary permissions for the lambda to communication to other services.
3. After gatekeeper-lambda function created shown in figure 5, the python code “gatekeeper_lambda.py” is zipped and uploaded to the function, mention the compatible python package as python 3.9.
4. Similarly created the business logic lambda “real_lambda” shown in figure 6, here the logic is created simply to return 200 status code once the lambda in invoked.
5. The required dependencies for the gatekeeper-lambda is attached to the function in the form of layer “crypt-layer” , so the layer is created shown in figure 7 with the zip file contained required dependencies installed.
6. Both lambda functions are created with required dependencies and deployed to be accessed.


aws
Search
[Option+S]
Europe (Ireland)
Account ID: 0406-7316-6587
deepak
Lambda > Functions > Create function
Basic information

Function name
Enter a name that describes the purpose of your function.
gatekeeper-lambda
Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime | Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.9

Architecture | Info
Choose the instruction set architecture you want for your function code.
arm64 [✓ x86_64]

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

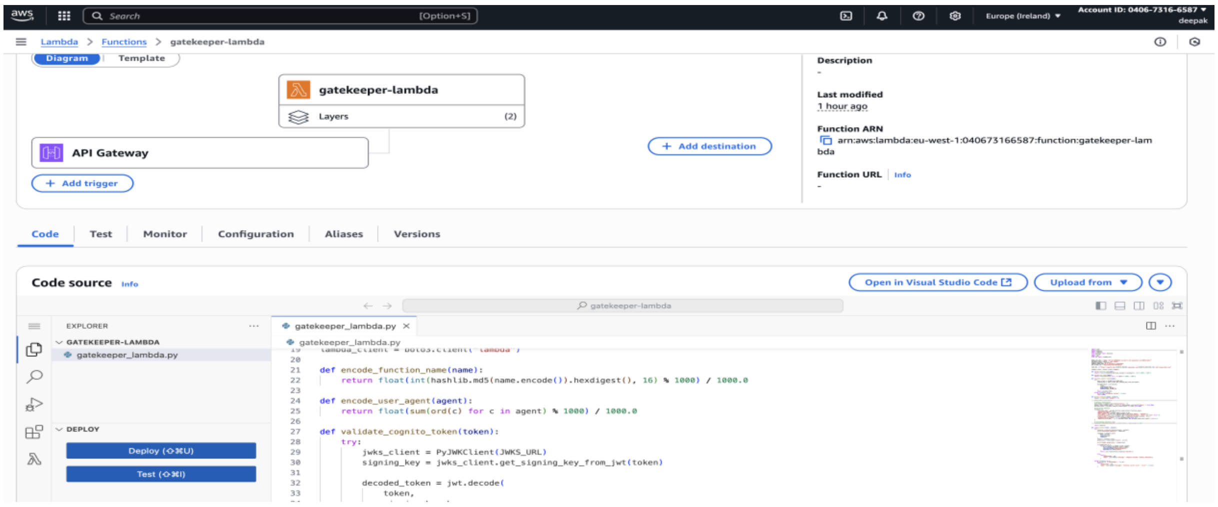
Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
Create a new role with basic Lambda permissions [✓ Use an existing role] Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
LambdaExecutionRole-ZeroTrust
View the LambdaExecutionRole-ZeroTrust role on the IAM console.

Additional configurations
Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

Figure 4: Lambda creation page



Description
-

Last modified
1 hour ago

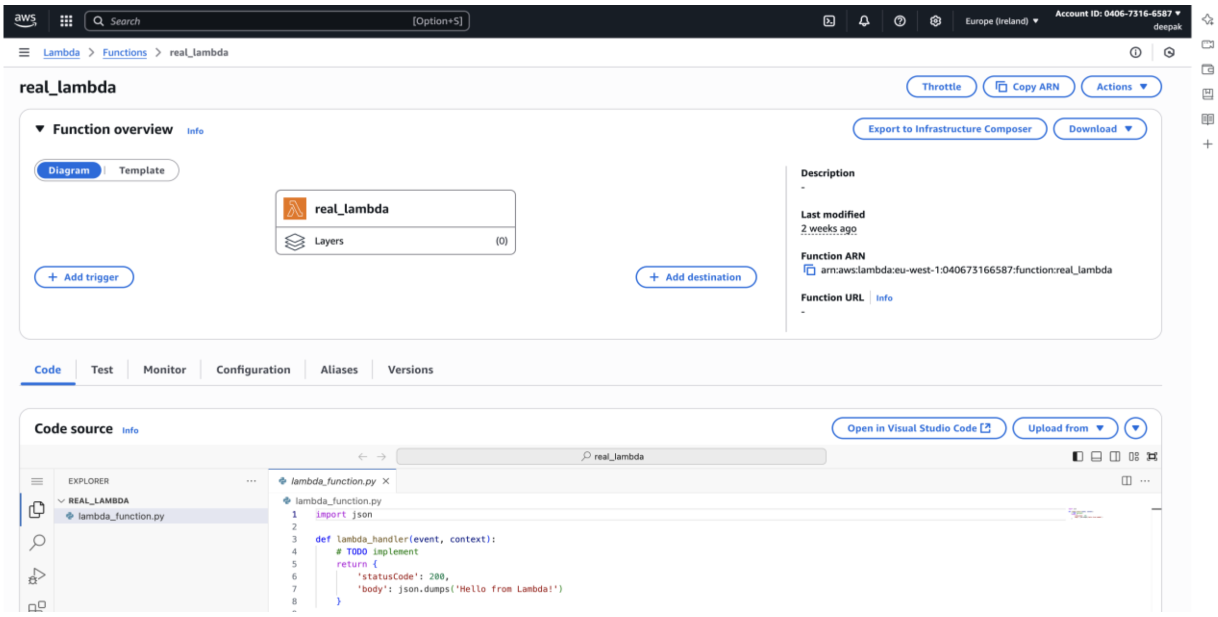
Function ARN
arn:aws:lambda:eu-west-1:040673166587:function:gatekeeper-lambda

Function URL Info
-

gatekeeper_lambda.py

```python
lambda_client = boto3.client("lambda")

def encode_function_name(name):
    return float(int(hashlib.md5(name.encode()).hexdigest(), 16) % 1000) / 1000.0

def encode_user_agent(agent):
    return float(sum(ord(c) for c in agent) % 1000) / 1000.0

def validate_cognito_token(token):
    try:
        jwks_client = PyJWKClient(JWKS_URL)
        signing_key = jwks_client.get_signing_key_from_jwt(token)

        decoded_token = jwt.decode(
            token,
```

Figure 5: Gatekeeper lambda



Description
-

Last modified
2 weeks ago

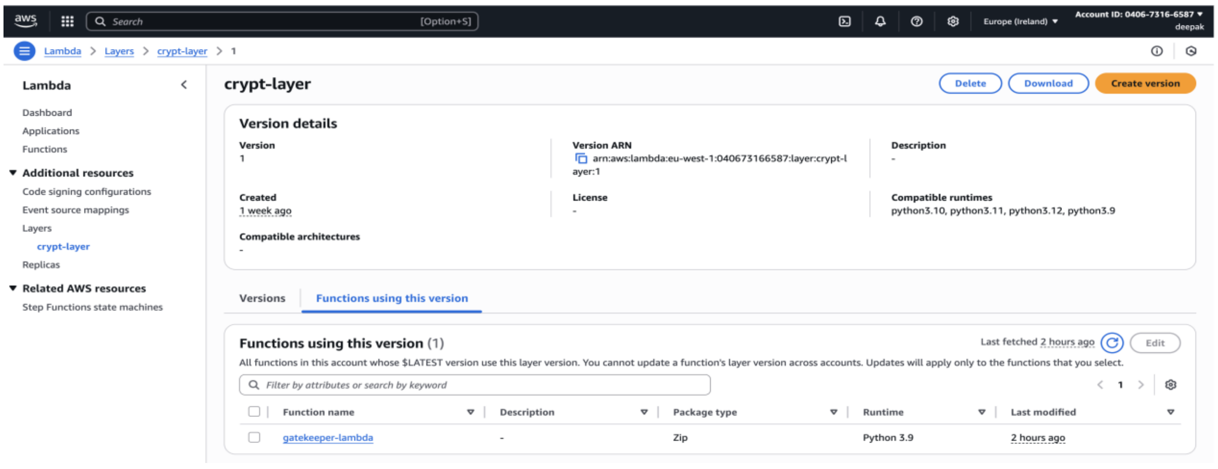
Function ARN
arn:aws:lambda:eu-west-1:040673166587:function:real_lambda

Function URL Info
-

lambda_function.py

```python
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

Figure 6: Business logic lambda



Version details

Version
1

Version ARN
arn:aws:lambda:eu-west-1:040673166587:layer:crypt-layer:1

Description
-

Created
1 week ago

License
-

Compatible runtimes
python3.10, python3.11, python3.12, python3.9

Compatible architectures
-

Functions using this version (1)

Last fetched 2 hours ago

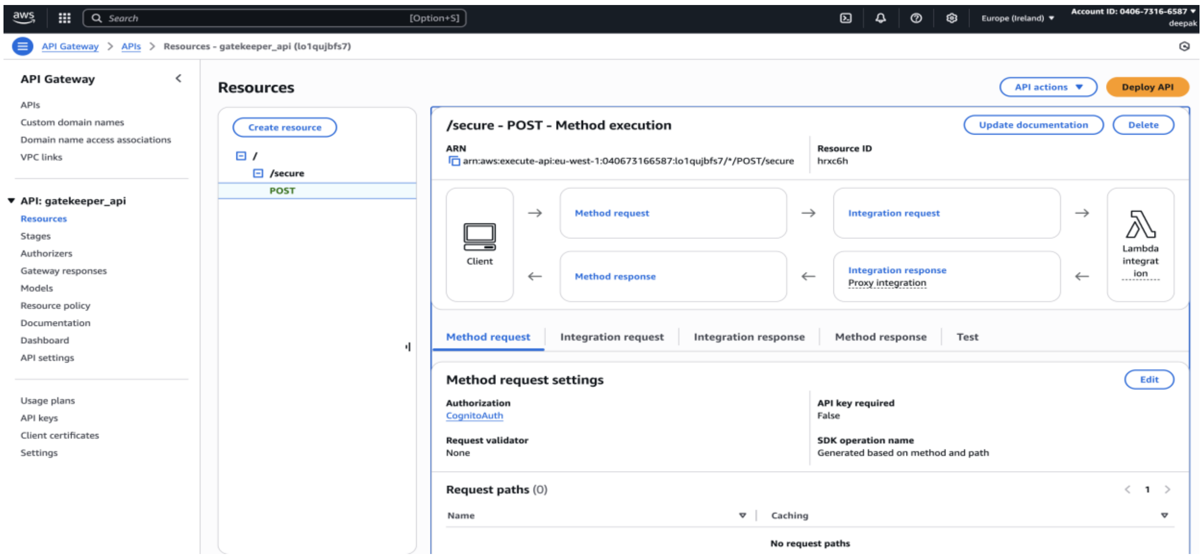
All functions in this account whose $LATEST version use this layer version. You cannot update a function's layer version across accounts. Updates will apply only to the functions that you select.

Function name	Description	Package type	Runtime	Last modified
gatekeeper-lambda	-	Zip	Python 3.9	2 hours ago

Figure 7: Lambda layers dashboard

API Gateway

1. In this project, REST API is created under API gateway for exposing the gatekeeper-lambda, in the API creation page provide the name for the API and create.
2. Once API is created, under the API create a resource with the path “/” and resource name “secure”.
3. Under the created resource, the method needs to be created. Here POST method is required so select POST method and provide the gatekeeper Lambda ARN for integration since the API invokes the lambda function.
4. Once method is created, the authorization needs to be attached, under method request attach “CognitoAuth” since we are using Cognito for user authentication. The figure 8 shows the all the configuration of API.



Account ID: 0406-7316-6587
deepak

API Gateway > APIs > Resources - gatekeeper_api (lo1qujbfs7)

API: gatekeeper_api

Resources

/
/secure
POST

/secure - POST - Method execution

ARN
arn:aws:execute-api:eu-west-1:040673166587:lo1qujbfs7/*/POST/secure

Resource ID
hrxc6h

Method request settings

Authorization
CognitoAuth

API key required
False

Request validator
None

SDK operation name
Generated based on method and path

Request paths (0)

Name	Caching

No request paths

Figure 8: API gateway resource page

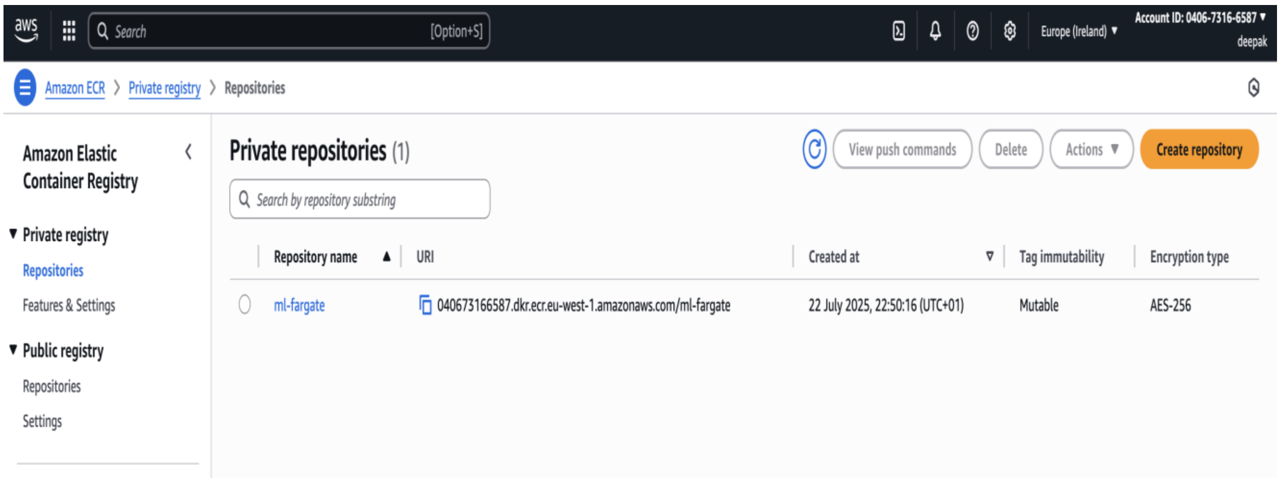
Docker and AWS Elastic Container Registry

1. The trained machine learning model (model.joblib) is containerize as Docker image, first a directory is created “ml-fargate” inside that placed Dockerfile, app.py, model.joblib and requirement.txt.
2. Once all the files placed inside the directory “ml-fargate” containerize as a docker image using below commands.
   ```
   $docker build -t ml-fargate .
   ```
3. The docker image can be run and tested locally using below command. Docker needs to be installed in the platform as prerequisites.
   ```
   $docker run -p 8080:8080 ml-fargate
   ```
4. Now, an ECR repository created using below AWS CLI command. AWS CLI should be installed in the platform as prerequisites.
   ```
   $ aws ecr create-repository --repository-name ml-fargate
   ```
5. Once the repository is created, docker needs to be authenticated to access ECR repository, using the below AWS CLI command.

```
$ aws ecr get-login-password --region eu-west-1 | \
docker login --username AWS --password-stdin 040673166587.dkr.ecr.eu-west-1.amazonaws.com
```

6. Now, tag the docker image as “latest” and push to the repository using below command.

```
$ docker tag ml-fargate 040673166587.dkr.ecr.eu-west-1.amazonaws.com/ml-fargate:latest
$ docker push 040673166587.dkr.ecr.eu-west-1.amazonaws.com/ml-fargate:latest
```

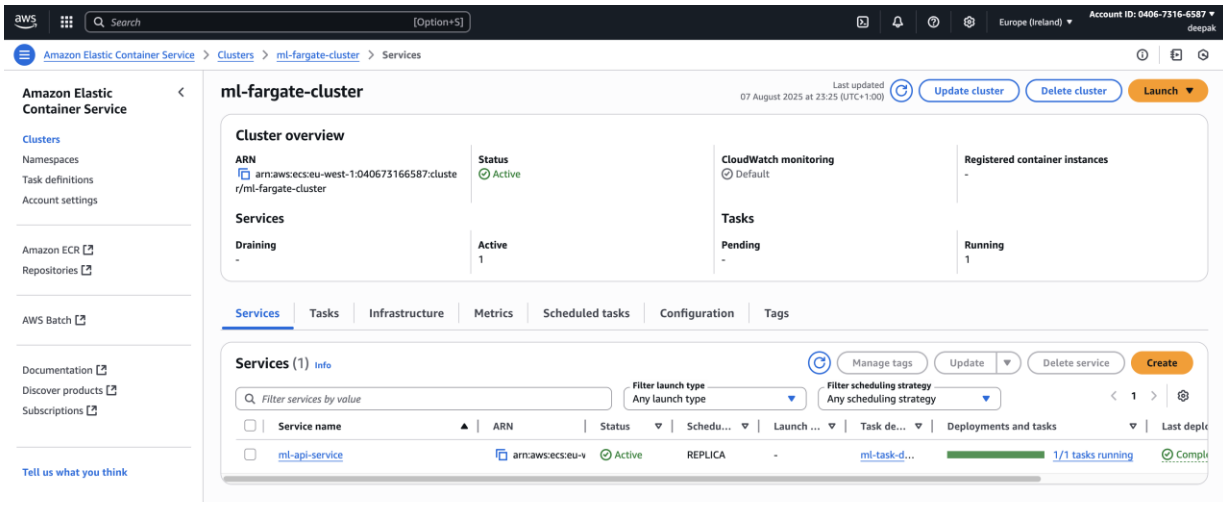


Repository name	URI	Created at	Tag immutability	Encryption type
ml-fargate	040673166587.dkr.ecr.eu-west-1.amazonaws.com/ml-fargate	22 July 2025, 22:50:16 (UTC+01)	Mutable	AES-256

Figure 9: AWS ECR repository

Amazon Elastic Container Service

1. To deploy the containerized docker image and to manage the docker container, AWS ECS is used. First, a cluster is created with Fargate (serverless) and with the cluster name “ml-fargate-cluster”, shown in figure 10.
2. Once cluster is created, a task definition is required. Under task definition create new, launch type as fargate, name as “ml-task-def” and choose role “ecsTaskExecutionRole” with necessary permission, shown in figure 11.
3. Under container definition, add container then select the docker image “040673166587.dkr.ecr.eu-west-1.amazonaws.com/ml-fargate:latest”, port mapping as “8080” click add and then create.
4. Now create a service under cluster, choose launch type ad fargate, then choose “ml-task-def”, then give the service name “ml-api-service” and number of task as “1”.
5. Now under networking section, provide the necessary network configuration, select the vpc and two subnets as required. Select a security group which has inbound rule for port “80” and “8080” open.
6. Enable ALB (Application Load Balancer), create new then give listener port as “8080” then create a target group with name “ml-api” and port “8080”.
7. Finally deploy the service and check the load balancer page for DNS, then test.
8. The DNS will be available in ALB page “ml-lb-1330515347.eu-west-1.elb.amazonaws.com”.



ml-fargate-cluster

Last updated 07 August 2025 at 23:25 (UTC+1:00)

Cluster overview

ARN	Status	CloudWatch monitoring	Registered container instances
arn:aws:ecs:eu-west-1:040673166587:cluster/ml-fargate-cluster	Active	Default	-

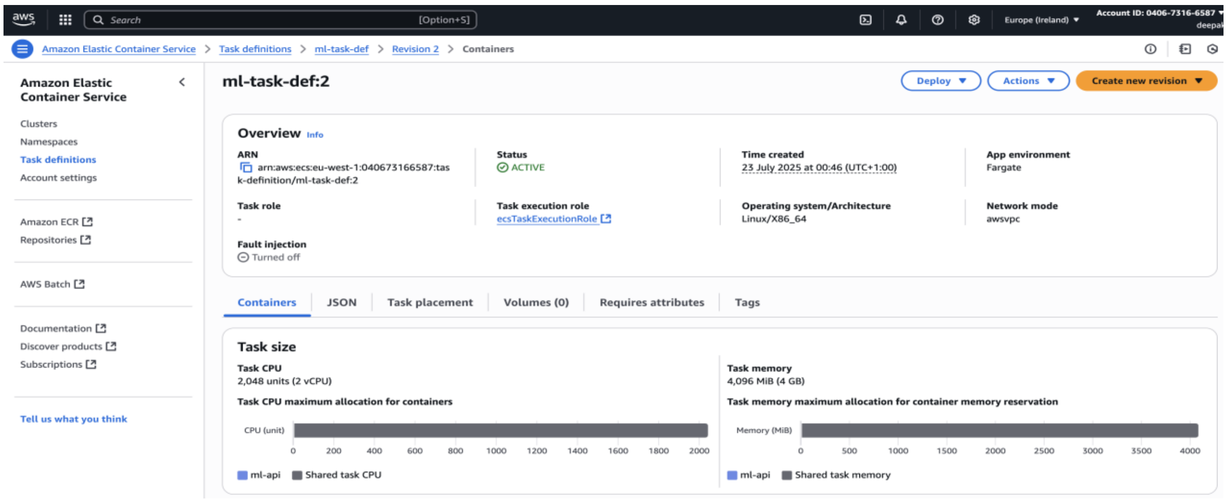
Services: Draining	Services: Active	Tasks: Pending	Tasks: Running
-	1	-	1

Services (1) Info

Filter launch type: Any launch type — Filter scheduling strategy: Any scheduling strategy

Service name	ARN	Status	Schedu...	Launch ...	Task de...	Deployments and tasks	Last deplo
ml-api-service	arn:aws:ecs:eu-v	Active	REPLICA	-	ml-task-d...	1/1 tasks running	Compl

Figure 10: AWS ECS cluster page



ml-task-def:2

Overview Info

ARN	Status	Time created	App environment
arn:aws:ecs:eu-west-1:040673166587:task-definition/ml-task-def:2	ACTIVE	23 July 2025 at 00:46 (UTC+1:00)	Fargate

Task role	Task execution role	Operating system/Architecture	Network mode
-	ecsTaskExecutionRole	Linux/X86_64	awsvpc

Fault injection: Turned off

Task size

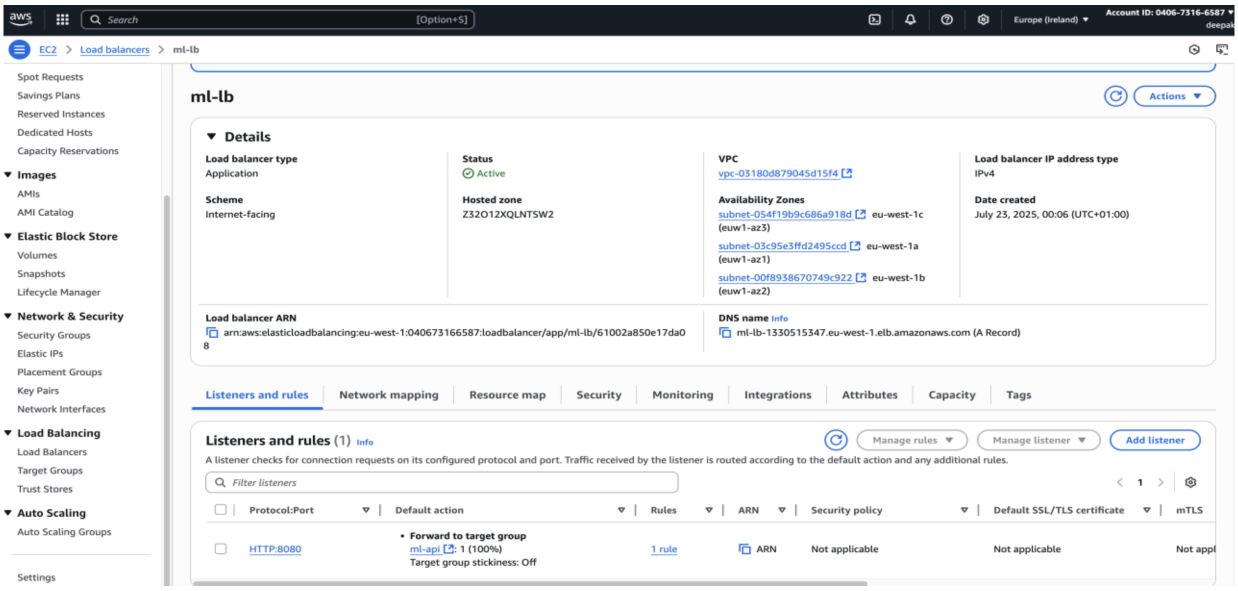
Task CPU: 2,048 units (2 vCPU)

Task CPU maximum allocation for containers

Task memory: 4,096 MiB (4 GB)

Task memory maximum allocation for container memory reservation

Figure 11: Task definition



ml-lb

Details

Load balancer type	Status	VPC	Load balancer IP address type
Application	Active	vpc-03180d879045d15f4	IPv4

Scheme	Hosted zone	Availability Zones	Date created
Internet-facing	Z32O12XQLNTSW2	subnet-054f19b9c686a918d eu-west-1c (euw1-az3); subnet-03c95e3ffd2495ccd eu-west-1a (euw1-az1); subnet-00f8938670749c922 eu-west-1b (euw1-az2)	July 23, 2025, 00:06 (UTC+01:00)

Load balancer ARN: arn:aws:elasticloadbalancing:eu-west-1:040673166587:loadbalancer/app/ml-lb/61002a850e17da08

DNS name Info: ml-lb-1330515347.eu-west-1.elb.amazonaws.com (A Record)

Listeners and rules (1) Info

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

Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS
HTTP:8080	Forward to target group ml-api: 1 (100%) Target group stickiness: Off	1 rule	ARN	Not applicable	Not applicable	Not appl

Figure 12: ALB dashboard

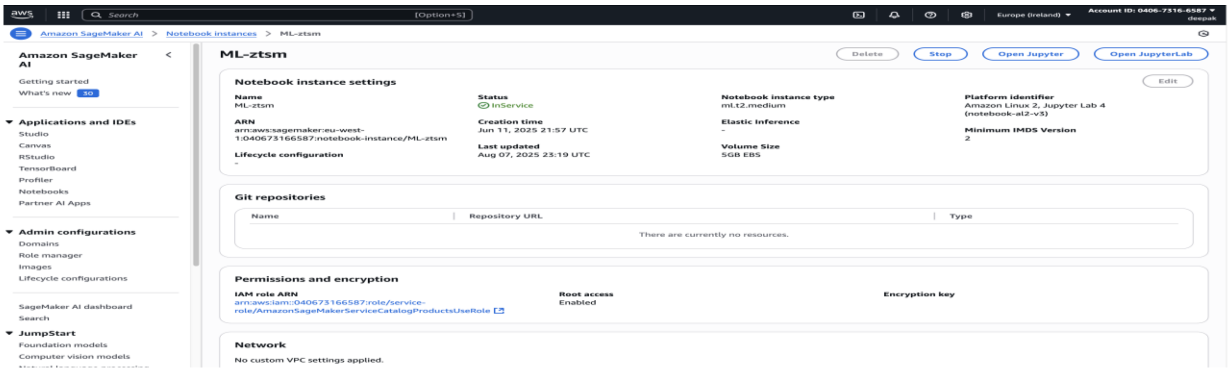
1.2 Anomaly Detection model training setup

This section covers the environment setup required for machine learning model training and storing the trained model.

AWS Sagemaker Jupyter notebook

1. To perform the machine learning modle training, a jupyter notebook is required. To create a notebook instance go to AWS Sagemaker AI service, then create a note book instance, provide the instance name, instance type and platform, also provide the IAM role with necessary permission.
2. Once the instance is created, open the jupyter notebook and upload the synthetically generated dataset "lambda_dataset.csv" and run the data preprocessing code to preprocess and clean the code for training, this will produce "features.csv" and "lables.csv"
3. Now train the model with training code and save the model as "model.joblib", before training the sklearn library version should be "1.2.1" it can be installed by below command.

   ```
   $pip install scikit-learn==1.2.1
   ```
4. Now the trained model is stored in the jupyter notebook instance and download for deployment.



ML-ztsm

Notebook instance settings

Name: ML-ztsm
Status: InService
Notebook instance type: ml.t2.medium
Platform identifier: Amazon Linux 2, Jupyter Lab 4 (notebook-al2-v3)
ARN: arn:aws:sagemaker:eu-west-1:040673166587:notebook-instance/ML-ztsm
Creation time: Jun 11, 2025 21:57 UTC
Elastic Inference: -
Minimum IMDS Version: 2
Last updated: Aug 07, 2025 23:19 UTC
Lifecycle configuration: -
Volume Size: 5GB EBS

Git repositories

Name	Repository URL	Type
There are currently no resources.		

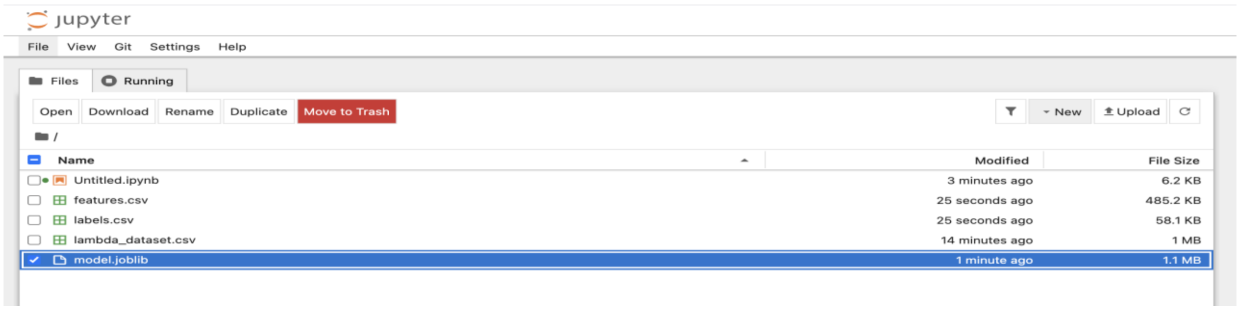
Permissions and encryption

IAM role ARN: arn:aws:iam::040673166587:role/service-role/AmazonSageMakerServiceCatalogProductsUseRole
Root access: Enabled
Encryption key:

Network

No custom VPC settings applied.

Figure 13: AWS Sagemaker instance detail



Name	Modified	File Size
Untitled.ipynb	3 minutes ago	6.2 KB
features.csv	25 seconds ago	485.2 KB
labels.csv	25 seconds ago	58.1 KB
lambda_dataset.csv	14 minutes ago	1 MB
model.joblib	1 minute ago	1.1 MB

Figure 14: Jupyter notebook instance

2 Testing

After the entire setup is done, the setup can be tested using below commands either in the local linux machine or ec2 instance.

1. Below command stores the value in variables

```
$ USER_POOL_CLIENT_ID="<userpool_id>"

$ USERNAME="<username>"

$ PASSWORD="<password>"

$ API_URL="<deployed api endpoint url>/secure"
```

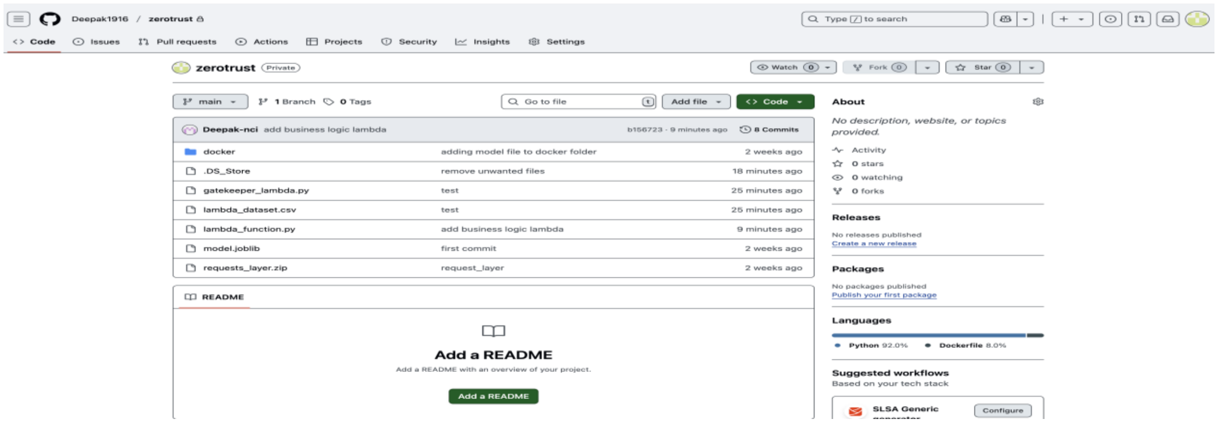
2. This command gets the cognito token.

```
$ TOKEN=$(aws cognito-idp initiate-auth \

  --auth-flow USER_PASSWORD_AUTH \

  --client-id $USER_POOL_CLIENT_ID \

  --auth-parameters
USERNAME=$USERNAME,PASSWORD=$PASSWORD \

  --query 'AuthenticationResult.IdToken' \

  --output text)
```

3. This command calls the gatekeeper lambda to validate the token and to call the model to predict.

```
$ curl -X POST "$API_URL" \

       -H "Authorization: Bearer $TOKEN" \

       -H "User-Agent: test-client" \

       -H "Content-Type: application/json" \

       -d '{"message": "hello"}'
```

3 GitHub Repository



Deepak-nci add business logic lambda b156723 · 9 minutes ago 8 Commits

docker	adding model file to docker folder	2 weeks ago
.DS_Store	remove unwanted files	18 minutes ago
gatekeeper_lambda.py	test	25 minutes ago
lambda_dataset.csv	test	25 minutes ago
lambda_function.py	add business logic lambda	9 minutes ago
model.joblib	first commit	2 weeks ago
requests_layer.zip	request_layer	2 weeks ago

README

Add a README

Add a README with an overview of your project.

Add a README

About

No description, website, or topics provided.

Activity
0 stars
0 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

Python 92.0% Dockerfile 8.0%

Suggested workflows
Based on your tech stack

SLSA Generic Configure

GitHub Repository URL: https://github.com/Deepak1916/zerotrust.git

References

Amazon Web Services. (2024). *Amazon API Gateway developer guide* [Documentation]. https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html

Amazon Web Services. (2024). *Amazon Cognito developer guide* [Documentation]. https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html

Amazon Web Services. (2024). *AWS Lambda developer guide* [Documentation]. https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

Amazon Web Services. (2024). *Amazon Elastic Container Service documentation* [Documentation]. https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html

Amazon Web Services. (2024). *Amazon Elastic Container Registry documentation* [Documentation]. https://docs.aws.amazon.com/AmazonECR/latest/userguide/what-is-ecr.html

Amazon Web Services. (2024). *Amazon SageMaker developer guide* [Documentation]. https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html