National College of Ireland

Configuration Manual: Optimizing Trust Score Algorithms in Zero Trust IAM Systems

Research Project
MSc Cloud Computing

Urvashi Kamlesh Sardare
Student ID: x22235248@student.ncirl.ie

School of Computing
National College of Ireland

Supervisor: Sai Emani

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Urvashi Kamlesh Sardare
Student ID:	x22235248@student.ncirl.ie
Programme:	MSc Cloud Computing
Year:	2025
Module:	Research Project
Supervisor:	Sai Emani
Submission Due Date:	11 August 2025
Project Title:	Configuration Manual: Optimizing Trust Score Algorithms in Zero Trust IAM Systems
Word Count:	1964
Page Count:	9

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	11 August 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.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

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

1 Introduction

1.1 Purpose of This Manual

This document provides a comprehensive, step-by-step guide for configuring and deploying the Multi-Cloud Zero Trust Identity and Access Management (IAM) system. The system is designed to operate in parallel across Amazon Web Services (AWS) and Google Cloud Platform (GCP), providing a dynamic, context-aware trust score to govern access requests.

This manual is intended for cloud administrators, DevOps engineers, and security professionals who are responsible for provisioning and maintaining cloud infrastructure. Following these instructions will result in a fully functional, serverless trust evaluation engine in both cloud environments.

1.2 System Overview

The system consists of two independent but conceptually aligned trust engines:

- **On AWS:** A serverless application using AWS Lambda, API Gateway, DynamoDB, and GuardDuty to calculate a trust score.
- **On GCP:** A parallel serverless application using Cloud Functions, Firestore, and Security Command Center to perform the same function.

Both engines are designed to be triggered by an API call containing user context, and they respond with a trust score and an access decision (Allow, MFA Required, or Deny).

1.3 Prerequisites

Before beginning the configuration, please ensure you have the following:

1. An AWS account with administrative privileges.
2. A GCP project with Owner or Editor privileges.
3. The AWS Command Line Interface (CLI) installed and configured with credentials for your AWS account.
4. The Google Cloud SDK (‘gcloud‘ CLI) installed and authenticated with your GCP account and project.
5. A local development environment with Python and the ‘zip‘ command-line utility.
6. All required source code files for the project:
    - For AWS: ‘lambda_function.py‘, ‘trust_score_algorithm.py‘
    - For GCP: ‘main.py‘, ‘trust_score_algorithm_gcp.py‘, ‘requirements.txt‘

Note: Throughout this manual, will need to replace placeholders like account IDs, ARNs, and project IDs with the actual values from the actual environments.

2 AWS Environment Configuration

This section details the setup of all necessary components within the AWS cloud.

2.1 IAM and S3 Initial Setup

First, create test users and an S3 bucket to simulate a realistic environment.

```
aws iam create-user --user-name user-lowtrust
aws iam create-user --user-name user-hightrust
```

Listing 1: Create IAM users for testing.

```
{
    "User": {
        "Path": "/",
        "UserName": "user-lowtrust",
        "UserId": "AIDAU3F5TO3AVZJHHM6F3",
        "Arn": "arn:aws:iam::333257602753:user/user-lowtrust",
        "CreateDate": "2025-07-16T17:00:22+00:00"
    }
}
```

Listing 2: Output for ‘user-lowtrust‘ creation.

```
aws s3api create-bucket --bucket zt-trust-data-bucket --region us-east
   -1
```

Listing 3: Create an S3 bucket for testing purposes.

```
{
    "Location": "/zt-trust-data-bucket"
}
```

Listing 4: Output for S3 bucket creation.

2.2 Lambda Function and Execution Role

The core of the system is a Lambda function with a dedicated IAM role.

2.2.1 Create the IAM Role

Create a file named ‘trust-policy.json‘.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Listing 5: Content of trust-policy.json.

Now, create the role.

```
aws iam create-role ^
  --role-name LambdaExecutionRole ^
  --assume-role-policy-document file://trust-policy.json
```

Listing 6: Create the Lambda execution role.

2.2.2 Attach Required Policies

```
aws iam attach-role-policy ^
  --role-name LambdaExecutionRole ^
  --policy-arn arn:aws:iam::aws:policy/service-role/
   AWSLambdaBasicExecutionRole

aws iam attach-role-policy ^
  --role-name LambdaExecutionRole ^
  --policy-arn arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess

aws iam attach-role-policy ^
  --role-name LambdaExecutionRole ^
  --policy-arn arn:aws:iam::aws:policy/AmazonGuardDutyReadOnlyAccess

aws iam attach-role-policy ^
  --role-name LambdaExecutionRole ^
  --policy-arn arn:aws:iam::aws:policy/CloudWatchFullAccess
```

Listing 7: Attach policies for Lambda, DynamoDB, and GuardDuty.

2.2.3 Package and Deploy the Lambda Function

```
zip function.zip lambda_function.py trust_score_algorithm.py
```

Listing 8: Package the Lambda source code.

```
# Replace the role ARN with the one generated for your account.
aws lambda create-function ^
  --function-name trust-evaluator ^
  --zip-file fileb://function.zip ^
  --handler lambda_function.lambda_handler ^
  --runtime python3.11 ^
  --role arn:aws:iam::333257602753:role/LambdaExecutionRole
```

Listing 9: Create the Lambda function.

2.3 API Gateway Integration

```
# The target ARN should be the ARN of the Lambda function created.
aws apigatewayv2 create-api ^
  --name TrustAPI ^
  --protocol-type HTTP ^
  --target arn:aws:lambda:us-east-1:333257602753:function:trust-
   evaluator
```

Listing 10: Create the HTTP API Gateway.

```
{
    "ApiEndpoint": "https://j6zil4ohei.execute-api.us-east-1.amazonaws.
   com",
    "ApiId": "j6zil4ohei",
    "ApiKeySelectionExpression": "$request.header.x-api-key",
    "CreatedDate": "2025-07-16T17:18:29+00:00",
    "DisableExecuteApiEndpoint": false,
    "Name": "TrustAPI",
    "ProtocolType": "HTTP",
    "RouteSelectionExpression": "$request.method $request.path"
}
```


Listing 11: Output from API Gateway creation, note the 'ApiEndpoint'.

2.4 Logging, Monitoring, and Security Setup

2.4.1 Create DynamoDB Table

```
aws dynamodb create-table ^
  --table-name trust_access_log ^
  --attribute-definitions AttributeName=username,AttributeType=S ^
  --key-schema AttributeName=username,KeyType=HASH ^
  --billing-mode PAY_PER_REQUEST
```

Listing 12: Create the DynamoDB table for access logs.

2.4.2 Enable GuardDuty

```
aws guardduty create-detector --enable
```

Listing 13: Enable Amazon GuardDuty.


```
{
    "DetectorId": "14cc0a29e1ec71b8124a4f63d77820b3"
}
```


Listing 14: Output from enabling GuardDuty.

2.4.3 Configure Alerting with SNS and CloudWatch

```
aws sns create-topic --name NotifyAdmin
```

Listing 15: Create an SNS topic for notifications.

```
# Replace the endpoint with your email address.
aws sns subscribe ^
  --topic-arn arn:aws:sns:us-east-1:333257602753:NotifyAdmin ^
  --protocol email ^
  --notification-endpoint usardare05@gmail.com
```

Listing 16: Subscribe an email address to the SNS topic.

```
# Replace the alarm-actions ARN with your SNS topic ARN.
aws cloudwatch put-metric-alarm ^
  --alarm-name LowTrustScoreDetected ^
  --metric-name TrustScore ^
  --namespace TrustIAM ^
  --statistic Average ^
  --threshold 0.4 ^
  --comparison-operator LessThanThreshold ^
  --evaluation-periods 1 ^
  --period 60 ^
  --alarm-actions arn:aws:sns:us-east-1:333257602753:NotifyAdmin
```

Listing 17: Create the CloudWatch alarm for low trust scores.

2.4.4 Update Lambda Function Code

After adding the logic for DynamoDB, GuardDuty, and CloudWatch metrics to your 'lambda$_{f}unction.py$', $updatethef unction$.

```
zip function.zip lambda_function.py trust_score_algorithm.py
aws lambda update-function-code ^
  --function-name trust-evaluator ^
  --zip-file fileb://function.zip
```

Listing 18: Update the Lambda function with new code.

3 GCP Environment Configuration

3.1 Initial Project and API Setup

```
gcloud config set project authentic-door-464610-h3
```

Listing 19: Set the active GCP project.

```
gcloud services enable ^
    cloudfunctions.googleapis.com ^
    firestore.googleapis.com ^
    iam.googleapis.com ^
    securitycenter.googleapis.com
```

Listing 20: Enable all required GCP APIs.

3.2 IAM Service Account Configuration

```
gcloud iam service-accounts create trust-engine-sa ^
  --description="Service Account for Trust Score Engine" ^
  --display-name="Trust Engine SA"
```

Listing 21: Create the Service Account for the trust engine.

3.3 Firestore Database Creation

```
gcloud firestore databases create --location=us-east1
```

Listing 22: Create a Firestore database in Native mode.

3.4 Cloud Function Deployment

```
# Replace the service account email with the one for your project.
gcloud functions deploy trust-evaluator ^
  --runtime python311 ^
  --trigger-http ^
  --allow-unauthenticated ^
  --entry-point trust_evaluator ^
  --source=. ^
  --region=us-east1 ^
  --service-account=trust-engine-sa@authentic-door-464610-h3.iam.
   gserviceaccount.com
```

Listing 23: Deploy the GCP Cloud Function.

3.5 Granting Permissions to the Service Account

```
# Permission to write to Firestore
gcloud projects add-iam-policy-binding authentic-door-464610-h3 ^
  --member="serviceAccount:trust-engine-sa@authentic-door-464610-h3.iam
   .gserviceaccount.com" ^
```

```
  --role="roles/datastore.user"

# Permission to read Security Command Center findings
gcloud projects add-iam-policy-binding authentic-door-464610-h3 ^
  --member="serviceAccount:trust-engine-sa@authentic-door-464610-h3.iam
    .gserviceaccount.com" ^
  --role="roles/securitycenter.findingsViewer"
```

Listing 24: Grant Firestore and SCC permissions.

4 System Verification

4.1 Testing the AWS Endpoint

```
# Replace the URL with your API Gateway endpoint.
curl -X POST https://j6zil4ohei.execute-api.us-east-1.amazonaws.com \
  -H "Content-Type: application/json" \
  -d '{
    "username": "user-lowtrust",
    "mfa_enabled": false,
    "geo_location": "India",
    "hour_of_day": 3,
    "access_history_score": 0.2
  }'
```

Listing 25: Test a low-trust scenario on AWS (no threat).

```
{"username": "user-lowtrust", "trust_score": 0.3, "decision": "Deny", "
   guardduty_alerts": false}
```

Listing 26: Expected output for low-trust scenario.

```
# Replace the detector-id with your own.
aws guardduty create-sample-findings ^
  --detector-id 14cc0a29e1ec71b8124a4f63d77820b3 ^
  --finding-types Recon:EC2/PortProbeUnprotectedPort
```

Listing 27: Simulate a GuardDuty finding.

```
# Use the same curl command as before.
curl -X POST https://j6zil4ohei.execute-api.us-east-1.amazonaws.com ...
```

Listing 28: Test low-trust scenario again (with threat).

```
{"username": "user-lowtrust", "trust_score": 0.0, "decision": "Deny", "
   guardduty_alerts": true}
```

Listing 29: Expected output with an active threat.

```
aws dynamodb get-item --table-name trust_access_log --key "{\"username\
   ": {\"S\": \"user-lowtrust\"}}"
```

Listing 30: Verify the log in DynamoDB.

```
{
    "Item": {
        "decision": { "S": "Deny" },
        "score": { "S": "0.0" },
        "username": { "S": "user-lowtrust" }
    }
}
```

Listing 31: Expected log entry in DynamoDB.

4.2 Testing the GCP Endpoint

```
# Replace the URL with your Cloud Function endpoint.
curl -X POST https://us-east1-authentic-door-464610-h3.cloudfunctions.net/trust-evaluator \
  -H "Content-Type: application/json" \
  -d '{
        "username": "user-lowtrust",
        "mfa_enabled": false,
        "geo_location": "India",
        "hour_of_day": 3,
        "access_history_score": 0.2
      }'
```

Listing 32: Test a low-trust scenario on GCP.

```
{"decision":"MFA Required","guardduty_alerts":false,"trust_score":0.4799...,"username":"user-lowtrust"}
```

Listing 33: Expected output for low-trust scenario on GCP.

5 Conclusion

By following the steps outlined in this manual, successfully deployed a sophisticated, serverless, multi-cloud Zero Trust IAM system. The configuration includes parallel trust evaluation engines on both AWS and GCP, complete with API endpoints, audit logging, and real-time threat intelligence integration.

The system is now fully operational and ready for further integration and testing. The deployed infrastructure provides a robust foundation for building more advanced security policies, integrating with identity providers, and tuning the trust score algorithms to meet specific organizational requirements. This manual serves as a complete and replicable record of the infrastructure setup, enabling consistent deployments across different environments.