

# **Telecom Resilience Platform: Serverless Billing, Fraud Detection & Disaster Recovery.**

**MUKESH GOND (B037) | JUBI RANKA (B039) |  
TANVI SHINDE (B042) | ISHA LAD (B045) | SHRIYA NAIR (B056)**



**First-Year M.Sc. Data Science Students, Nilkamal School of Mathematics,  
Applied Statistics & Analytics (NSOMASA), SVKM's NMIMS  
(Deemed-to-be) University, Mumbai, Maharashtra, India**

**MENTOR :**

**Prof. Pratik Desai**

**Nilkamal School of Mathematics, Applied Statistics & Analytics (NSOMASA),  
SVKM's NMIMS (Deemed-to-be) University, Mumbai, Maharashtra, India**

# TABLE OF CONTENTS

TOPIC	PAGE NO.
Executive Summary	3
Business Use Case	3
Solution Overview	4
AWS Architecture	5
Implementation Plan	6
Security And Compliance Considerations	38
Expected Outcomes	38
Conclusion	39

# Executive Summary

The telecom industry requires a scalable, highly available, and secure platform to monitor real-time network usage, automate customer billing, prevent fraud, and provide a self-service portal for customers. This document outlines the AWS-based architecture and detailed implementation plan for such a system, ensuring operational efficiency and cost optimization.

---

## Business Use Case

### Challenges Faced by Telecom Companies

1. **Real-Time Call & Data Usage Monitoring** – Telecom companies need to track millions of call logs, data usage, and network health metrics in real-time to ensure seamless operations and customer satisfaction.
2. **Automated Customer Billing** – Manual billing processes can lead to errors and delays; thus, a highly available automated billing system ensures customers are charged correctly and on time.
3. **Fraud Detection & Prevention** – Suspicious activity such as SIM cloning, spam calls, and fraudulent transactions must be detected and mitigated in real-time to protect users and revenue.
4. **Customer Self-Service Portal** – A user-friendly web portal enables customers to manage their accounts, recharge their plans, and resolve queries without needing customer support intervention.
5. **Disaster Recovery & Failover** – Ensuring service continuity during unexpected outages is crucial, requiring robust backup, failover, and disaster recovery mechanisms.

# Solution Overview

## Uniqueness of the Solution

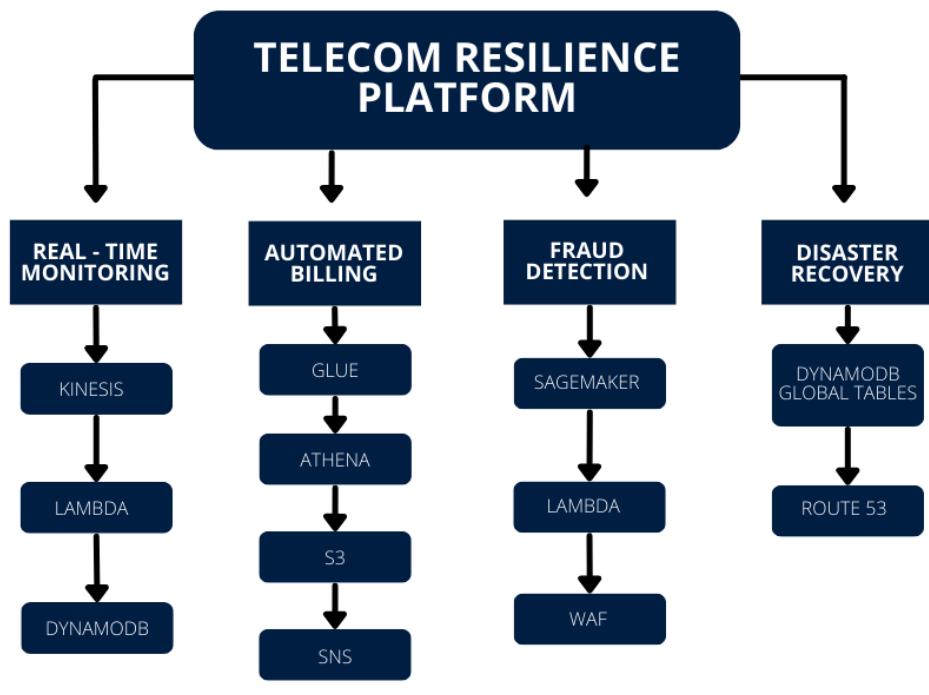
- **Serverless & Scalable** – Optimized performance with AWS Lambda, Kinesis, and DynamoDB, eliminating the need for maintaining physical infrastructure.
- **Real-Time Processing** – Instant tracking and monitoring using AWS streaming services ensures up-to-date network insights.
- **AI-Driven Fraud Prevention** – Amazon SageMaker provides predictive analysis and fraud detection, reducing security risks.
- **Disaster-Resilient** – Multi-region replication and automated backups ensure business continuity even in system failures.

## Proposed AWS Services

- **Data Ingestion & Processing:** Amazon Kinesis, AWS Lambda, AWS Step Functions.
- **Data Storage & Management:** Amazon DynamoDB, Amazon S3, Amazon Glacier.
- **Security & Compliance:** AWS WAF, AWS Shield, AWS IAM.
- **Billing & Fraud Detection:** AWS Glue, Amazon SageMaker, Amazon Athena.
- **Customer Self-Service:** Amazon API Gateway, Amazon Cognito, CloudFront.
- **Disaster Recovery:** AWS Backup, Multi-Region Replication, AWS Route 53.

# AWS Architecture

## Complete System Architecture Diagram



## CUSTOMER SELF SERVICE PORTAL



# Implementation Plan

## Step 1: Setting Up Real-Time Data Monitoring

1. **Create Amazon Kinesis Data Stream** – Configure Kinesis Data Streams to ingest real-time telecom data, such as call logs and network usage.
2. **Deploy AWS Lambda Functions** – Write serverless Lambda functions to process and filter the streaming data for further analysis.
3. **Store Data in Amazon DynamoDB** – Store processed data in a NoSQL DynamoDB table for efficient querying and retrieval.
4. **Enable Amazon CloudWatch Monitoring** – Set up CloudWatch to track system performance and alert on anomalies.

## Step 2: Implementing Automated Billing System

1. **Extract Usage Data with AWS Glue** – Use AWS Glue to crawl and extract call and data usage details from DynamoDB and logs.
2. **Analyze Billing Data with Amazon Athena** – Query structured billing data using Amazon Athena to generate invoices dynamically.
3. **Store Generated Invoices in Amazon S3** – Store customer invoices securely in Amazon S3 for easy access and retrieval.
4. **Notify Customers via Amazon SNS** – Send automated SMS and email notifications to customers with billing summaries.

## Step 3: Deploying Fraud Detection System

1. **Stream Call Metadata Using Kinesis Firehose** – Collect call patterns and metadata to detect suspicious behavior.
2. **Train AI Fraud Models with Amazon SageMaker** – Use historical fraud data to train machine learning models that identify fraudulent transactions.
3. **Trigger AWS Lambda for Instant Fraud Prevention** – Deploy Lambda functions to block fraudulent transactions in real-time.
4. **Implement AWS WAF for Threat Mitigation** – Use AWS WAF to prevent bot attacks, spam calls, and brute-force attempts.

## **Step 4: Developing Customer Self-Service Portal**

1. **Host Portal on Amazon S3 & CloudFront** – Deploy a static web application on S3 with CloudFront for global content delivery.
2. **Use Amazon API Gateway for Service Requests** – Implement REST APIs to handle user account management, plan recharges, and support requests.
3. **Enable User Authentication with Amazon Cognito** – Secure customer accounts with multi-factor authentication and OAuth2 login.
4. **Store Customer Preferences in DynamoDB** – Store customer preferences and historical transactions in a scalable database.

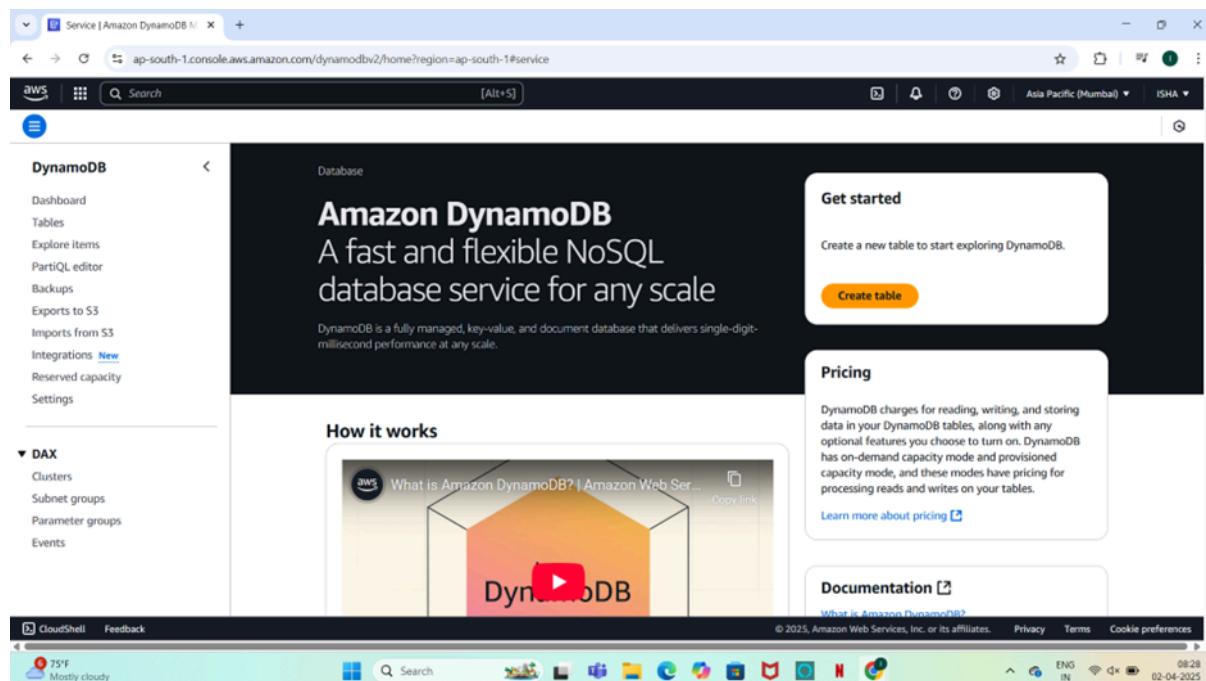
## **Step 5: Configuring Disaster Recovery & Failover**

1. **Enable AWS Backup for Regular Snapshots** – Schedule automated snapshots of databases and storage to prevent data loss.

2. **Set Up Multi-Region Replication for DynamoDB & S3** – Ensure high availability by replicating data across different AWS regions.
3. **Use AWS Route 53 for Automatic Failover** – Configure Route 53 to switch traffic to backup servers in case of an outage.
4. **Conduct Disaster Recovery Drills Regularly** – Perform scheduled failover tests to validate the resilience of the system.

## Step-by-Step Implementation

### Step 1: Set Up DynamoDB Global Tables



#### Primary Region (Mumbai/ap-south-1)

Since Mumbai is chosen as the primary region, all data will be first written here. The setup ensures high availability and quick access for users in this region.

1. Create DynamoDB Table: The table stores application data that needs to be replicated across multiple regions.
  - Table name: MultiRegionTable
  - Partition key: id (String) -

- The **partition key** uniquely identifies each item in the table.
  - It ensures that data is distributed efficiently for fast lookups.
  - Choosing **id** as a string makes it flexible for different types of data entries.
- Enable Point-in-Time Recovery (PITR)
    - Why PITR?
    - PITR allows automatic backups of data **every second** for up to **35 days**.
    - In case of accidental deletion or corruption, you can restore data from any point in time.
    - This adds an extra layer of **data protection** and helps recover from human errors or system failures.

The screenshot shows the AWS DynamoDB console interface. On the left, there's a navigation sidebar with options like Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. Below that is a section for DAX Clusters, Subnet groups, Parameter groups, and Events. The main content area is titled "MultiRegionTable". At the top, there are tabs for Settings, Indexes, Monitor, Global tables (which is selected), Backups, Exports and streams, and Permissions. Under the "Global tables" tab, there's a section for "Replicas (1) info". It says "Other AWS Regions to which you have replicated this table. You are using global tables version 2019.11.21." There's a table with columns: Replication Region, Status, Capacity mode, and Endpoint. One row is shown: "Asia Pacific (Singapore) ap-southeast-1" with "Active" status, "On-demand" capacity mode, and the endpoint "dynamodb.ap-southeast-1.amazonaws.com". There are "Delete replica" and "Create replica" buttons at the top of this section. At the bottom of the main content area, there are links for CloudShell, Feedback, and a footer with copyright information: "© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".

The screenshot shows the AWS DynamoDB console interface. On the left, there's a navigation sidebar with options like Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. Under the Tables section, 'Tables' is selected. The main area shows a confirmation message: 'The MultiRegionTable table was created successfully.' Below this, a table titled 'Tables (1) Info' lists one table: MultiRegionTable. The table details are as follows:

Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read capacity mode
MultiRegionTable	Active	id (\$)	-	0	0	Off	☆	On-demand

This screenshot is identical to the one above, showing the AWS DynamoDB console with the same navigation sidebar and the creation of the MultiRegionTable. The confirmation message and the table details table are also identical.

The screenshot shows the AWS DynamoDB console with the URL [ap-south-1.console.aws.amazon.com/dynamodbv2/home?region=ap-south-1#tables](https://ap-south-1.console.aws.amazon.com/dynamodbv2/home?region=ap-south-1#tables). The page displays a success message: "The MultiRegionTable table was created successfully." Below this, the "Tables (1) Info" section shows a table named "MultiRegionTable" with the following details:

Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read capacity mode
MultiRegionTable	Active	id (\$)	-	0	0	Off		On-demand

This screenshot is identical to the one above, showing the AWS DynamoDB console with the same URL and success message. The "Tables (1) Info" section shows the "MultiRegionTable" with the same details as the first screenshot.

## 1. Enable Global Tables:

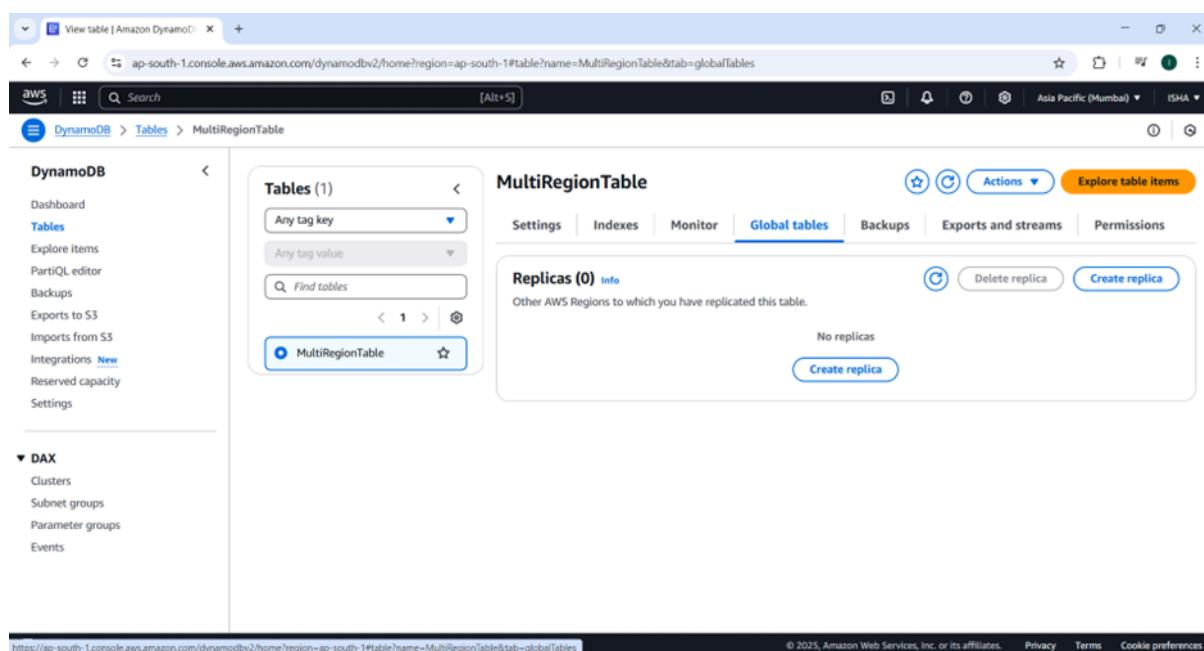
Why do we need Global Tables?

- If Mumbai (ap-south-1) faces an outage, data should still be available in another AWS region (Singapore in this case).

- Global Tables automatically replicate data across multiple AWS regions to keep backups updated in real-time.
- It ensures low-latency access for users in different geographic locations.

### Adding Replica Region: Singapore (ap-southeast-1)

- Singapore is chosen as the **backup region** to ensure that data remains accessible even if Mumbai goes down.
- This enables a **multi-region disaster recovery setup**, where users can switch to Singapore if needed.
- Go to Global Tables → Add replica region: Singapore (ap-southeast-1).
- Wait ~10 mins for replication to activate.



The screenshot shows the Amazon DynamoDB console with the URL <https://ap-south-1.console.aws.amazon.com/dynamodbv2/home?region=ap-south-1#table?name=MultiRegionTable&tab=globalTables>. The left sidebar shows 'Tables' selected under 'DynamoDB'. The main content area displays the 'MultiRegionTable' settings, specifically the 'Global tables' tab. It lists one table, 'MultiRegionTable', with no replicas. There are buttons for 'Delete replica' and 'Create replica'.

The screenshot shows the 'Create replica' dialog box. It includes sections for 'Replication settings' (Current Region: Asia Pacific (Mumbai)), 'Available replication Regions' (Asia Pacific (Singapore) ap-southeast-1 selected), and 'IAM role' (AWSServiceRoleForDynamoDBReplication selected). A note at the bottom states: 'For replication to work, DynamoDB Streams will be activated automatically for new and old images.' There are 'Cancel' and 'Create replica' buttons at the bottom right.

[Clear Shell](#) [Feedback](#) © 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

The screenshot shows the AWS Lambda console interface. On the left, the navigation bar includes 'Dashboard', 'Tables' (selected), 'Explore items', 'PartiQL editor', 'Backups', 'Exports to S3', 'Imports from S3', 'Integrations' (New), 'Reserved capacity', and 'Settings'. Below this is a section for 'DAX' with options like 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The main content area is titled 'MultiRegionTable' under the 'Global tables' tab. It displays a table with one row for 'Replicas (1)'. The row shows 'Replication Region' as 'Asia Pacific (Mumbai) ap-south-1', 'Status' as 'Active', 'Capacity mode' as 'On-demand', and 'Endpoint' as 'dynamodb.ap-south-1.amazonaws.com'. There are buttons for 'Actions' (with 'Delete replica' and 'Create replica') and 'Explore table items'.

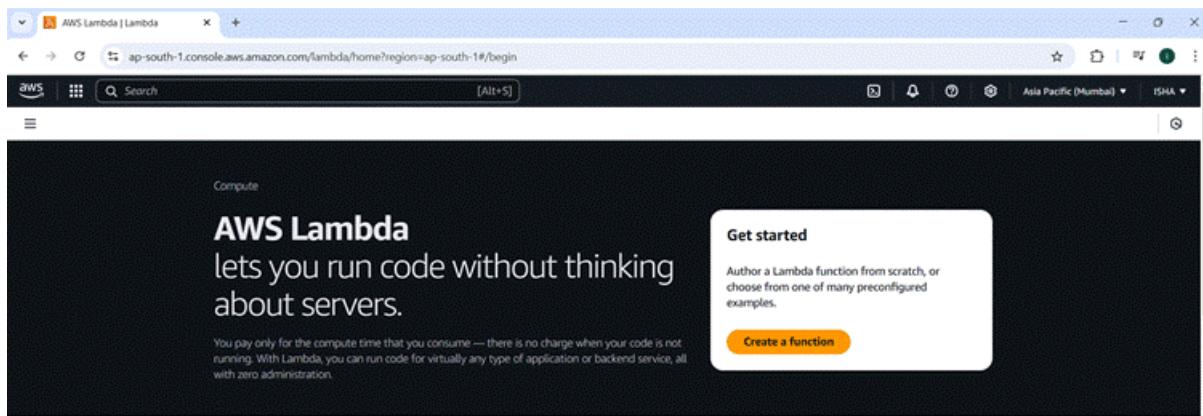
This screenshot is identical to the one above, showing the MultiRegionTable configuration in the Asia Pacific (Mumbai) region. The table 'Replicas (1)' lists the same information: Replication Region (Asia Pacific (Mumbai) ap-south-1), Status (Active), Capacity mode (On-demand), and Endpoint (dynamodb.ap-south-1.amazonaws.com). The interface includes the same navigation bar and DAX section on the left, and the same 'Actions' and 'Explore table items' buttons on the right.

## Step 2: Deploy Lambda Functions

Primary Region (Mumbai)

### 1. Create Lambda Function:

- This function automatically writes data to DynamoDB in the Mumbai region (ap-south-1).
- Helps store important data so that it can be used later.



- Name: PrimaryRegionWriter
- Runtime: Python 3.12

The screenshot shows the AWS Lambda console interface. At the top, a message says "Successfully created the function PrimaryRegionWriter. You can now change its code and configuration. To invoke your function with a test event, choose 'Test'." Below this, the "PrimaryRegionWriter" function is listed with a "Layers" section showing "(0)". There are buttons for "Throttle", "Copy ARN", and "Actions". The "Code" tab is selected, showing a "Code source" section with an "Upload from" button. The sidebar on the right is titled "Tutorials" and includes a section for "Create a simple web app" with a "Start tutorial" button.

- IAM Role:

- Attach **AmazonDynamoDBFullAccess** (for testing): Allows the function to store data in DynamoDB.
- Attach **AWSLambdaBasicExecutionRole** (for logs): Allows logging and monitoring in CloudWatch (helps in debugging).

The screenshot shows the AWS IAM console. On the left, the navigation menu includes "Identity and Access Management (IAM)", "Access management", "Access reports", and "CloudTrail". The main panel displays the details of the role "PrimaryRegionWriter-role-gze47y34". It shows the creation date as April 02, 2025, at 08:47 (UTC+05:30). The ARN is listed as arn:aws:iam:977099003849:role/service-role/PrimaryRegionWriter-role-gze47y34. Under the "Permissions" tab, there is one attached policy: "AWSLambdaBasicExecutionRole-d1692f93-...". The "Permissions policies" section shows a table with one item: "AWSLambdaBasicExecutionRole-d1692f93-...". The "Permissions boundary (not set)" and "Generate policy based on CloudTrail events" sections are also visible.

The screenshot shows the 'Add permissions' dialog in the AWS IAM console. The 'Current permissions policies (1)' section is collapsed. The 'Other permissions policies (2/1043)' section is expanded, showing a list of policies. One policy, 'AWSLambdaBasicExecutionRole', is selected and highlighted in blue. The 'Description' for this policy states: 'Provides write permissions to CloudWatch Metrics'. At the bottom right of the dialog are 'Cancel' and 'Add permissions' buttons.

The screenshot shows the 'PrimaryRegionWriter-role-gze47y34' role details page in the AWS IAM console. A green success message at the top states: 'Policies have been successfully attached to role.' Below this, the 'Permissions' tab is selected, showing three attached policies: 'AmazonDynamoDBFullAccess', 'AWSLambdaBasicExecutionRole', and 'AWSLambdaBasicExecutionRole-d1692f93...'. The 'Attached entities' column shows that each policy is attached to 1 entity. At the bottom right of the page are 'Simulate', 'Remove', and 'Add permissions' buttons.

## 1. Lambda Code:

- Uses boto3 (AWS SDK for Python) to connect to DynamoDB in Mumbai.
- Generates a unique ID using uuid and saves a message to the table.
- Returns a success message confirming the data is stored.

```

import boto3

import uuid

def lambda_handler(event, context):

    dynamodb=boto3.resource('dynamodb',region_name='ap-south-1')

    table = dynamodb.Table('MultiRegionTable')

    item_id = str(uuid.uuid4())

    table.put_item(Item={

        'id': item_id,

        'region': 'ap-south-1',

        'message': 'Data from Mumbai Lambda'

    })

    return {

        'statusCode': 200,

        'body': f'Success! Inserted item {item_id} into DynamoDB'

    }

```

## 2. Add API Gateway Trigger:

- **Purpose:** Allows external applications to trigger the Lambda function via an **HTTP request**.
- **Why?** So users or other systems can **send data to Mumbai's database** through a simple API call.

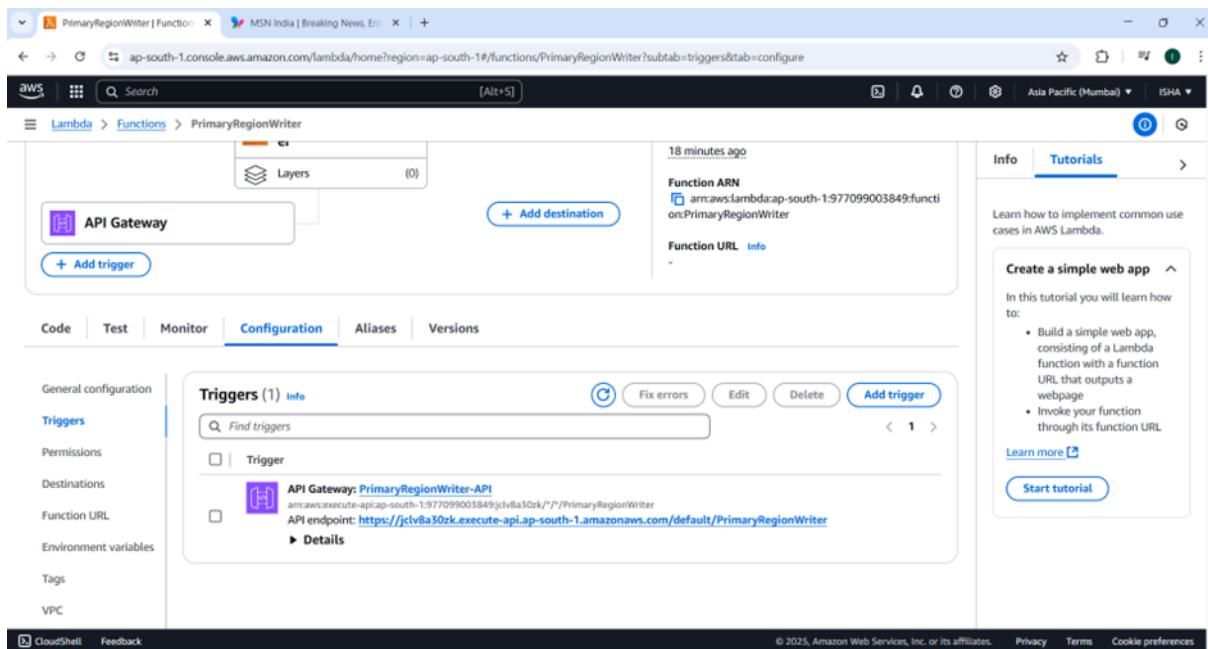
## API Type: HTTP API

### Endpoint:

[https://\[api-id\].execute-api.ap-south-1.amazonaws.com/default/PrimaryRegionWriter](https://[api-id].execute-api.ap-south-1.amazonaws.com/default/PrimaryRegionWriter)

The screenshot shows the 'Add trigger' configuration page for an AWS Lambda function. The 'Trigger configuration' section is open, showing the 'API Gateway' tab selected. Under 'Intent', the 'Create a new API' option is selected. Under 'API type', the 'HTTP API' option is selected. The 'Security' section shows 'Open' as the chosen security mechanism. The right sidebar contains a 'Tutorials' section titled 'Create a simple web app' with a 'Start tutorial' button.

The screenshot shows the 'Functions' page for an AWS Lambda function named 'PrimaryRegionWriter'. The 'Function overview' section is active, showing the function's ARN and a 'Function URL' field. The 'Configuration' tab is selected at the bottom. The right sidebar contains a 'Tutorials' section titled 'Create a simple web app' with a 'Start tutorial' button.



## Secondary Region (Singapore)

This acts as a backup in case the Mumbai region fails.

The function works the same way but stores data in DynamoDB in Singapore (ap-southeast-1) instead.

The DynamoDB connection is updated to point to Singapore (ap-southeast-1).

This ensures the data is stored in the secondary region for disaster recovery.

### 1. Repeat the same steps with:

- Lambda Name: SecondaryRegionWriter
- Region: ap-southeast-1
- Code Update:

```
dynamodb      =      boto3.resource('dynamodb',
region_name='ap-southeast-1')
```

The screenshot shows the 'Create function' wizard in the AWS Lambda console. The top navigation bar includes tabs for 'Info' and 'Tutorials'. The 'Tutorials' tab is selected, displaying a 'Create a simple web app' section with a brief description and two bullet points: 'Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage' and 'Invoke your function through its function URL'. Below this is a 'Learn more' link and a 'Start tutorial' button.

**Create function** [Info](#)

Choose one of the following options to create your function.

Author from scratch  
Start with a simple Hello World example.

Use a blueprint  
Build a Lambda application from sample code and configuration presets for common use cases.

Container image  
Select a container image to deploy for your function.

**Basic information**

**Function name**  
Enter a name that describes the purpose of your function.

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 when writing your function. Note that the console code editor supports only Node.js, Python, and Ruby.  
 [Edit](#)

**Architecture** [Info](#)  
Choose the instruction set architecture you want for your function code.  
 x86\_64  
 arm64

**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](#)

[CloudShell](#) [Feedback](#) © 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

The screenshot shows the 'SecondaryRegionWriter' function details page. The top navigation bar includes tabs for 'Info' and 'Tutorials'. The 'Tutorials' tab is selected, displaying a 'Create a simple web app' section with a brief description and two bullet points: 'Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage' and 'Invoke your function through its function URL'. Below this is a 'Learn more' link and a 'Start tutorial' button.

**SecondaryRegionWriter**

**Function overview** [Info](#)

[Diagram](#) [Template](#)

**SecondaryRegionWriter**

[Layers](#) (0)

[+ Add trigger](#) [+ Add destination](#)

[Throttle](#) [Copy ARN](#) [Actions](#)

[Export to Infrastructure Composer](#) [Download](#)

**Description**  
-

**Last modified**  
2 seconds ago

**Function ARN**  
[arn:aws:lambda:ap-southeast-1:977099003849:function:SecondaryRegionWriter](#)

**Function URL** [Info](#)  
-

[Code](#) [Test](#) [Monitor](#) [Configuration](#) [Aliases](#) [Versions](#)

**Code source** [Info](#)

[Upload from](#) [Download](#)

[CloudShell](#) [Feedback](#) © 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

SecondaryRegionWriter | Func. x MSN India | Breaking News, En... x +

ap-southeast-1.console.aws.amazon.com/lambda/home?region=ap-southeast-1#/functions/SecondaryRegionWriter?newFunction=true&tab=code

Lambda > Functions > SecondaryRegionWriter

Successfully updated the function SecondaryRegionWriter.

**Code source** [Info](#)

**EXPLORER**

**SECONDARYREGIONWRITER**

**lambda\_function.py**

```

lambda_function.py
1 import boto3
2 import uuid
3
4 def lambda_handler(event, context):
5     dynamodb = boto3.resource('dynamodb', region_name='ap-southeast-1') # Updated region
6     table = dynamodb.Table('MultiRegionTable')
7
8     item_id = str(uuid.uuid4())
9     table.put_item(Item={
10         'id': item_id,
11         'region': 'ap-southeast-1',
12         'message': 'Data written from Singapore Lambda'
13     })
14
15     return {
16         'statusCode': 200,
17         'body': f'Success! Inserted item {item_id} into DynamoDB (Singapore)'
18     }

```

**DEPLOY**

Deploy (Ctrl+Shift+U)

test (Ctrl+Shift+I)

**TEST EVENTS [NONE SELECTED]**

Create new test event

CloudShell Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

SecondaryRegionWriter | Func. x MSN India | Breaking News, En... x +

ap-southeast-1.console.aws.amazon.com/lambda/home?region=ap-southeast-1#/functions/SecondaryRegionWriter?newFunction=true&tab=configure

Lambda > Functions > SecondaryRegionWriter

Successfully updated the function SecondaryRegionWriter.

**Code** **Test** **Monitor** **Configuration** [Aliases](#) [Versions](#)

**General configuration** [Info](#)

Description	Memory	Ephemeral storage
-	128 MB	512 MB
Timeout	SnapStart <a href="#">Info</a>	
0 min 3 sec	None	

[Edit](#)

Triggers

Permissions

Destinations

Function URL

Environment variables

Tags

VPC

RDS databases

Monitoring and operations tools

Concurrency and recursion detection

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

The screenshot shows the AWS IAM Roles page. On the left, a sidebar navigation includes 'Identity and Access Management (IAM)', 'Access management' (with 'Roles' selected), and 'Access reports'. The main content area displays the 'SecondaryRegionWriter-role-itn6rbfs' role. It shows a 'Summary' section with creation date (April 02, 2025, 09:17 UTC+05:30), last activity (none), ARN (arn:aws:iam::977099003849:role/service-role/SecondaryRegionWriter-role-itn6rbfs), and maximum session duration (1 hour). Below this is a 'Permissions' tab, which lists one managed policy: 'AWSLambdaBasicExecutionRole-1a6f2c53...'. A 'Permissions boundary' section is shown as '(not set)'. At the bottom right, there are links for 'Edit', 'Delete', 'Simulate', 'Remove', and 'Add permissions'.

The screenshot shows the 'Add permissions' dialog for the 'SecondaryRegionWriter-role-itn6rbfs' role. It lists 'Current permissions policies (1)' and 'Other permissions policies (2/1044)'. Under 'Other permissions policies', a search bar filters results by 'AWSLambdaBasicExecutionRole'. One policy, 'AWSLambdaBasicExecutionRole', is selected and highlighted with a blue border. Other policies listed include 'AWSLambdaBasicExecutionRole-11c54ed9-4895-4309-b499-7fd78d5...', 'AWSLambdaBasicExecutionRole-a79b91e6-0e90-4748-8941-f0e795b...', and 'AWSLambdaBasicExecutionRole-d1692f93-c0b5-4611-b537-48d9062...'. A 'Description' column for the selected policy states: 'Provides write permissions to CloudWa...'. At the bottom right of the dialog are 'Cancel' and 'Add permissions' buttons.

The screenshot shows the AWS IAM Roles page. The role name is SecondaryRegionWriter-role-itn6rbfs. It was created on April 02, 2025, at 09:17 (UTC+05:30). The ARN is arn:aws:iam:977099003849:role/service-role/SecondaryRegionWriter-role-itn6rbfs. The maximum session duration is 1 hour. Three managed policies are attached: AmazonDynamoDBFullAccess and AWSLambdaBasicExecutionRole. The permissions tab is selected.

The screenshot shows the AWS Lambda Functions page. The function name is SecondaryRegionWriter. The function ARN is arn:aws:lambda:ap-southeast-1:977099003849:function:SecondaryRegionWriter. The function URL is https://ap-southeast-1.execute-api.amazonaws.com/default/SecondaryRegionWriter. The function was last modified 9 minutes ago. The code source tab is selected, showing the code editor with the file SecondaryRegionWriter.js.

The screenshot shows the AWS Lambda console interface. The top navigation bar includes tabs for SecondaryRegionWriter, PrimaryHealthC, SecondaryRegionW, SecondaryRegionW, Roles | IAM | Global, AWSLambdaBasicD, MSN India | Breakin, and others. The main content area is titled "SecondaryRegionWriter". Under "Function overview", there is a diagram showing the function name "SecondaryRegionWriter" connected to an "API Gateway" trigger. Below the diagram are buttons for "Add destination" and "Add trigger". The "Configuration" tab is selected, showing sections for "General configuration" and "Execution role". The "Execution role" section displays the ARN `arn:aws:lambda:ap-southeast-1:977099003849:function:SecondaryRegionWriter` and the function URL `https://12345678901234567890123456789012.execute-api.ap-southeast-1.amazonaws.com`. A sidebar on the right provides a tutorial on creating a simple web app.

The screenshot shows the AWS Lambda console interface, similar to the previous one but with the "Test" tab selected. The "Test event" section is active, showing a "Create new event" button and a "TestPermissions" input field. The "Event sharing settings" section has the "Private" radio button selected. The "Template - optional" section contains the text "Hello World". The "Event JSON" section is visible at the bottom. The browser status bar indicates NIFTY +0.12%.

The screenshot shows the AWS Lambda console with the URL [ap-southeast-1.console.aws.amazon.com/lambda/home?region=ap-southeast-1#/functions/SecondaryRegionWriter?subtab=permissions&tab=testing](https://ap-southeast-1.console.aws.amazon.com/lambda/home?region=ap-southeast-1#/functions/SecondaryRegionWriter?subtab=permissions&tab=testing). The page displays a success message: "The test event 'TestPermissions' was successfully saved." Below this, there's a "Test event" section with "Info" and "Delete" buttons, and a "Test" button highlighted in orange. A "CloudWatch Logs Live Tail" button is also present. The "Event name" field contains "TestPermissions". The "Event JSON" section shows the following JSON code:

```
1 * []
2 "key1": "value1",
3 "key2": "value2",
4 "key3": "value3"
5 []
```

This screenshot is identical to the one above, but the "Test" tab is selected in the navigation bar. The "Code" tab is now grayed out. The rest of the interface, including the success message, event configuration, and sidebar, remains the same.

The screenshot shows the AWS Lambda console with the function `SecondaryRegionWriter`. A green banner at the top indicates that the test event "TestPermissions" was successfully saved. The main area displays the execution logs:

```
{
  "statusCode": 200,
  "body": "Success! Inserted item 8a3480a2-0a26-4d06-a5b2-8d018d12eb8a into DynamoDB (Singapore)"
}
```

Below the logs, there's a summary section with various metrics:

- Code SHA-256:** O9zjFee2eEx/L55VKwFMXIQbMpcyNc/ON7CyTJPZTOU=
- Function version:** \$LATEST
- Duration:** 2374.42 ms
- Resources configured:** 128 MB
- Init duration:** 287.19 ms
- Execution time:** 33 seconds ago
- Request ID:** adc17a5c-de0c-4607-b354-a82e0x8f57a3
- Billed duration:** 2375 ms
- Max memory used:** 80 MB

On the right side, there's a sidebar with a "Tutorials" tab selected, showing a "Create a simple web app" tutorial.

The screenshot shows the AWS Lambda console with the function `SecondaryRegionWriter` and the "Monitor" tab selected. A green banner at the top indicates that the test event "TestPermissions" was successfully saved.

The "Monitor" section includes several links: View CloudWatch logs, View Application Signals, View X-Ray traces, View Lambda Insights, and View CodeGuru profiles. It also features a "Filter metrics by" dropdown set to "Function".

Under "CloudWatch metrics", it says: "Lambda sends runtime metrics for your functions to Amazon CloudWatch. The metrics shown are an aggregate view of all function runtime activity. To view metrics for the unqualified or \$LATEST resource, choose Filter by. To view metrics for a specific function version or alias, choose Aliases or Versions, select the alias or version, and then choose Monitor."

The main area displays three metrics cards:

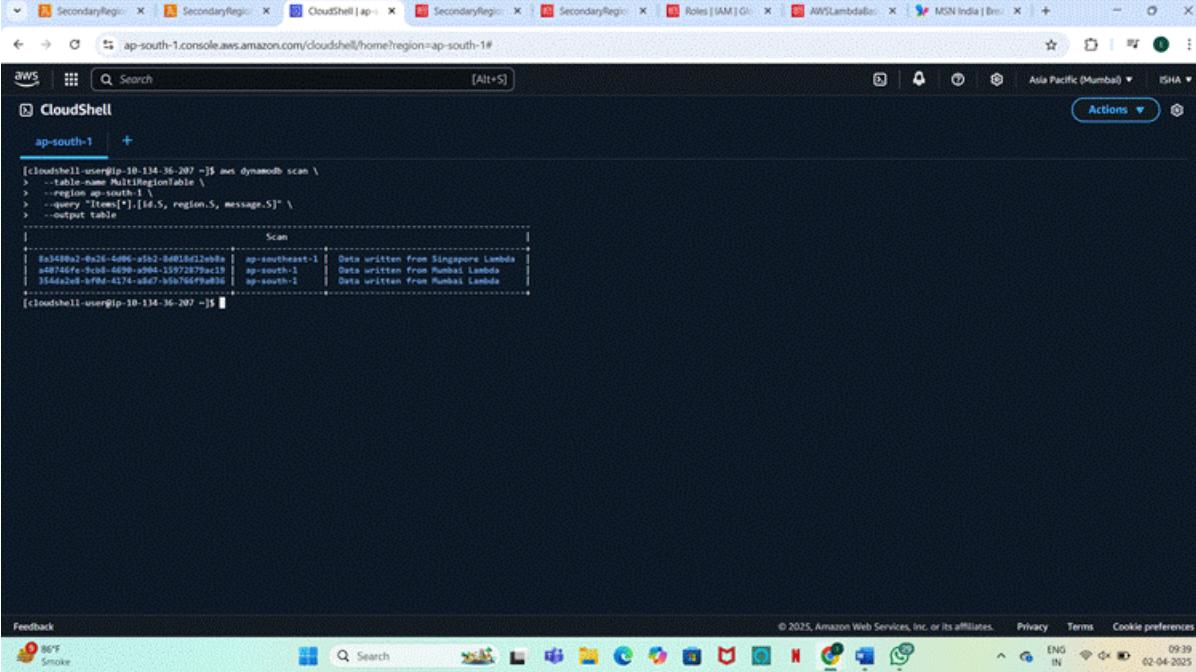
- Invocations:** Count 2
- Duration:** Milliseconds 2.37k
- Error count and success rate (%):** Count 1, Success Rate 100%

At the bottom, the URL is https://ap-southeast-1.console.aws.amazon.com/lambda/home?region=ap-southeast-1#functions/SecondaryRegionWriter?tab=monitoring

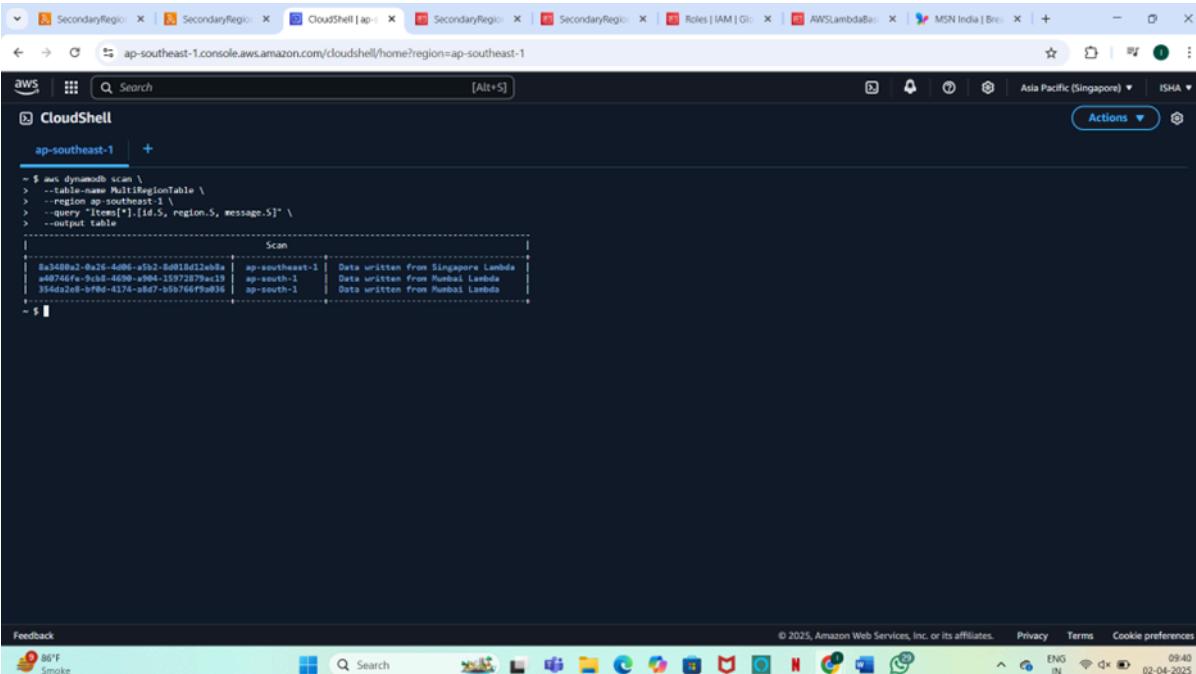
The screenshot shows the AWS CloudWatch Log streams page. On the left, the navigation menu includes CloudWatch, Favorites and recent, Dashboards, Alarms, Logs (selected), Metrics, X-Ray traces, Events, Application Signals, Network Monitoring, and Insights. Under Logs, Log groups (New) is selected. The main content area displays a log stream named "2025/04/02/[SLATEST]d11370896df64e2694dff884a34404a9". The log stream details include ARN, Creation time (2 minutes ago), Retention (Never expire), and Stored bytes (0). The Log streams tab is active, showing one log stream entry. The entry timestamp is 2025-04-02 04:01:38 (UTC).

The screenshot shows the AWS CloudWatch Log events page. The navigation menu is identical to the previous screenshot. The main content area displays log events for the same log stream. The log events table has columns for Timestamp and Message. The messages show runtime details like INIT\_START, START RequestId, END RequestId, and REPORT RequestId. The timestamp for the first event is 2025-04-02T04:01:36.106Z.

## Step 3: Configure Route 53 Health Checks



```
[cloudshell-user@ip-10-134-36-207 ~]$ aws dynamodb scan \
> --table-name MultiRegionTable \
> --region ap-south-1 \
> --query "Items[*].[id,S,region,S,message,S]" \
> --output table
|
| Scan |
|-----|
| 8a3480a2-0x26-4d86-a5b2-0d018d12eb98 | ap-southeast-1 | Data written from Singapore Lambda |
| a0f74dfe-9c08-4690-a9b4-15972879e219 | ap-south-1 | Data written from Mumbai Lambda |
| 354da2e8-bf0d-4174-a8d7-b5b76649b036 | ap-south-1 | Data written from Mumbai Lambda |
|
[cloudshell-user@ip-10-134-36-207 ~]$
```



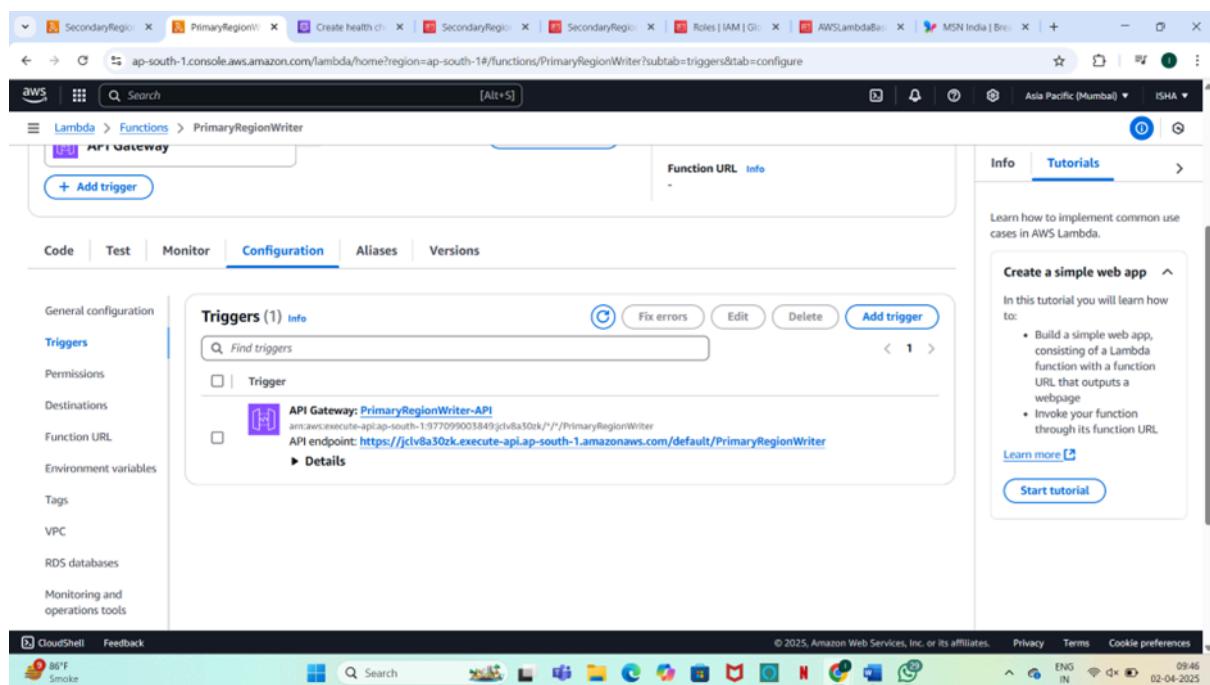
```
$ aws dynamodb scan \
> --table-name MultiRegionTable \
> --region ap-southeast-1 \
> --query "Items[*].[id,S,region,S,message,S]" \
> --output table
|
| Scan |
|-----|
| 8a3480a2-0x26-4d86-a5b2-0d018d12eb98 | ap-southeast-1 | Data written from Singapore Lambda |
| a0f74dfe-9c08-4690-a9b4-15972879e219 | ap-south-1 | Data written from Mumbai Lambda |
| 354da2e8-bf0d-4174-a8d7-b5b76649b036 | ap-south-1 | Data written from Mumbai Lambda |
|
$
```

### 1. Create Health Check for Mumbai:

**Health checks monitor the status of our APIs in different AWS regions (Mumbai & Singapore).**

**They help Route 53 automatically switch traffic to a backup region if the primary region fails.**

- **Name:** PrimaryHealthCheck-ap-south-1
- **Domain:**  
[api-id].execute-api.ap-south-1.amazonaws.com
- **Path:** /default/PrimaryRegionWriter
- **Protocol:** HTTPS



## 1. Create Health Check for Singapore:

- **Name: SecondaryHealthCheck-ap-southeast-1**
- **Domain:**  
**[api-id].execute-api.ap-southeast-1.amazonaws.com**
- **Path: /default/SecondaryRegionWriter**

**Create health check** Info

Create a new health check to monitor the health of a specified resource.

**Configuration**

Configuration options have cost implications. Pricing will vary based on the selected resource type. Advanced configurations have an additional cost. [View pricing](#)

**Name - optional**

A friendly name that lets you easily find a health check in the dashboard.

**SecondaryHealthCheck-Singapore**

The name must have 1-256 characters, Valid characters: A-Z, a-z, 0-9, - (hyphen), and \_ (underscore).

**Resource**

**Endpoint**  
Establish a connection with the resource to determine its health status.

**Calculated health check**  
The status of the health check is based on the status of the other health checks.

**CloudWatch alarm**  
The status of the health check is based on the state of a specified CloudWatch alarm.

**Specify endpoint by**

IP address

Domain name

**Domain name**

The path can be any value for which your endpoint will return an HTTP status code of 2xx or 3xx when the endpoint is healthy.

HTTP

**Advanced configuration**

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences ENG IN 09:59 02-04-2025

**Route 53**

- Dashboard
- Hosted zones
- Health checks**
- Profiles [New](#)
- IP-based routing
- Traffic flow
- Domains
- Resolver

**SecondaryHealthCheck-Singapore** Info

Health check with id 57183d14-a789-46ad-942b-9a47a56d12f1 has been created successfully.

**Configuration**

ID <a href="#">57183d14-a789-46ad-942b-9a47a56d12f1</a>	URL <a href="http://tahjxh427e.execute-api.ap-southeast-1.amazonaws.com:80/">http://tahjxh427e.execute-api.ap-southeast-1.amazonaws.com:80/</a>	Specified endpoint by Domain name
State <a href="#">Enabled</a>	Status <a href="#">Unknown</a>	Inverted No

**Advanced configuration**

**Metrics** [Metrics Info](#) [View metrics in CloudWatch](#)

**Health check status**

3h 1d 1w UTC timezone [Edit](#)

SENSEK +0.54% CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences ENG IN 10:00 02-04-2025

## Verify Health Checks:

Wait ~5 mins → Check status in Route 53.

The screenshot shows the AWS Route 53 Health checks console. On the left, there's a navigation sidebar with options like Route 53, Dashboard, Hosted zones, Health checks (selected), Profiles (New), IP-based routing, Traffic flow, Domains, Resolver, VPCs, Inbound endpoints, Outbound endpoints, Rules, and Query logging. The main area is titled "Health checks (2)" and contains a table with two rows:

ID	Name	Details	Status in last 24 hours	Current s...	Alarm	State
0bx26492-5052-4...	PrimaryHealthC...	http://jclv8a30...	<span style="color: red;">Unhealthy</span>	None, ...	<span style="color: green;">Enabled</span>	
57183d14-a789-4...	SecondaryHealt...	http://tahjxd42...	<span style="color: green;">Healthy</span>	None, ...	<span style="color: green;">Enabled</span>	

At the bottom of the page, there are links for CloudShell, Feedback, and various browser tabs. The status bar at the bottom right shows "© 2025, Amazon Web Services, Inc. or its affiliates.", "Privacy", "Terms", "Cookie preferences", "CloudWatch Metrics", "SENSEX +0.54%", and the date "02-04-2025".

## Step 4: Set Up Route 53 Failover

1. Go to Route 53 → Hosted Zones → Select your domain.
2. Create A Record:

- **Name: api.yourdomain.com**
- **Type: A → Alias to API Gateway (Mumbai).**
- **Routing Policy: Failover**
- **Primary: Linked to Mumbai health check.**
- **Secondary: Linked to Singapore health check.**

The screenshot shows the AWS Route 53 Health checks console. On the left, there's a navigation sidebar with options like Route 53, Dashboard, Hosted zones, Health checks (which is selected), Profiles (New), IP-based routing, CIDR collections, Traffic flow, Traffic policies, Policy records, Domains, Registered domains, Pending requests, and Resolver. The main area is titled "Health checks (2) Info" and contains a table with two rows of data:

ID	Name	Details	Status in last 24 hours	Current s...	Alarm	State
0bx26492-5052-4a...	PrimaryHealthC...	http://jclv8a30z...	<span style="color: red;">Unhealthy</span>	None, ...	<span style="color: green;">Enabled</span>	
57183d14-a789-4...	SecondaryHealt...	http://tahjxh42...	<span style="color: red;">Unhealthy</span>	None, ...	<span style="color: green;">Enabled</span>	

A blue banner at the top says "Introducing the new Route 53 health checks console experience. We've redesigned the health checks console to make it easier to use. The changes include a new layout for faster access to information. Learn more about the changes and let us know what you think. Or you can use the old console." There's also a "Create health check" button.

## Step 5: Test Failover

### 1. Simulate Mumbai Failure:

- Throttle Mumbai Lambda (Reserved Concurrency = 0).

### 2. Observe:

- Route 53 shifts traffic to Singapore within ~2 mins.

### 3. Restore Mumbai:

- Set concurrency back to Unreserved.

### 4. Troubleshooting

- Health Check "Unhealthy"
- No Data Replication
- API Gateway 403/5XX

```
| 354da2a8-bf8d-4174-a8d7-b5b766f9e036 | ap-south-1 | Data written from Mumbai Lambda |
+-----+-----+-----+
[aws] SecondaryRegion X [aws] SecondaryRegion X [aws] Health check X [CloudShell] X [aws] SecondaryRegion X [aws] SecondaryRegion X [aws] Roles IAM X [aws] AWSLambda X [aws] MSN India X + - X X
← → ⌂ ap-south-1.console.aws.amazon.com/cloudshell/home?region=ap-south-1
aws CloudShell [Alt+S] Actions
CloudShell
ap-south-1 +
| 354da2a8-bf8d-4174-a8d7-b5b766f9e036 | ap-south-1 | Data written from Mumbai Lambda |
+-----+-----+-----+
[aws] curl -v https://jclvba30zk.execute-api.ap-south-1.amazonaws.com/default/PrimaryRegionWriter
Host: jclvba30zk.execute-api.ap-south-1.amazonaws.com was resolved.
IPV6: (none)
IPV4: 11.232.145.37, 35.154.254.127
Trying 11.232.145.37:443...
Connected to jclvba30zk.execute-api.ap-south-1.amazonaws.com (11.232.145.37) port 443
ALPN: curl offers h2,http/1.1
TLSv1.3 (OUT), TLS handshake, Client hello (1):
[Alpn: /etc/pki/tls/certs/ca-bundle.crt]
[AlgHash: none]
[EncType: ECDHE-RSA-AES128-GCM-SHA256]
TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
TLSv1.3 (IN), TLS handshake, Certificate (11):
TLSv1.3 (IN), TLS handshake, CERT verify (15):
TLSv1.3 (IN), TLS handshake, FINISHED (20):
TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
TLSv1.3 (OUT), TLS handshake, Finished (20):
SSL connection using TLSv1.3 / TLS_AES_128_GCM_SHA256 / X25519 / RSASSA_PSS
ALPN: server accepted h2
Server certificate:
subject: CN=*.execute-api.ap-south-1.amazonaws.com
start date: Aug 30 00:00:00 2024 GMT
expire date: Sep 28 23:59:59 2025 GMT
subjectAltName: host "jclvba30zk.execute-api.ap-south-1.amazonaws.com" matched cert's "*.execute-api.ap-south-1.amazonaws.com"
issuer: C=US; O=Amazon RSA 2048 M2
SSL certificate verify ok
Certificate level 0: Public key type RSA (2048/112 Bits/secBits), signed using sha256WithRSAEncryption
Certificate level 1: Public key type RSA (2048/112 Bits/secBits), signed using sha256WithRSAEncryption
Certificate level 2: Public key type RSA (2048/112 Bits/secBits), signed using sha256WithRSAEncryption
using HTTP/2
[HTTP/2] [1] OPENED stream for https://jclvba30zk.execute-api.ap-south-1.amazonaws.com/default/PrimaryRegionWriter
[HTTP/2] [1] [:method: GET]
[HTTP/2] [1] [:scheme: https]
[HTTP/2] [1] [:path: /default/PrimaryRegionWriter]
[HTTP/2] [1] [:path: /default/PrimaryRegionWriter]
[HTTP/2] [1] [:user-agent: curl/1.8.5.0]
[HTTP/2] [1] [:accept: */*]
> GET /default/PrimaryRegionWriter HTTP/2
Feedback
22 min delay
Western Express... Search Privacy Terms Cookie preferences
© 2025, Amazon Web Services, Inc. or its affiliates. ENG IN 10:06 02-04-2025
```

```
| 354da2a8-bf8d-4174-a8d7-b5b766f9e036 | ap-south-1 | Data written from Mumbai Lambda |
+-----+-----+-----+
[aws] SecondaryRegion X [aws] SecondaryRegion X [aws] Health check X [CloudShell] X [aws] SecondaryRegion X [aws] SecondaryRegion X [aws] Roles IAM X [aws] AWSLambda X [aws] MSN India X + - X X
← → ⌂ ap-south-1.console.aws.amazon.com/cloudshell/home?region=ap-south-1
aws CloudShell [Alt+S] Actions
CloudShell
ap-south-1 +
| 354da2a8-bf8d-4174-a8d7-b5b766f9e036 | ap-south-1 | Data written from Mumbai Lambda |
+-----+-----+-----+
[aws] curl -v https://jclvba30zk.execute-api.ap-south-1.amazonaws.com/default/PrimaryRegionWriter
Host: jclvba30zk.execute-api.ap-south-1.amazonaws.com was resolved.
IPV6: (none)
IPV4: 11.232.145.37, 35.154.254.127
Trying 11.232.145.37:443...
Connected to jclvba30zk.execute-api.ap-south-1.amazonaws.com (11.232.145.37) port 443
ALPN: curl offers h2,http/1.1
TLSv1.3 (OUT), TLS handshake, Client hello (1):
[Alpn: /etc/pki/tls/certs/ca-bundle.crt]
[AlgHash: none]
[EncType: ECDHE-RSA-AES128-GCM-SHA256]
TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
TLSv1.3 (IN), TLS handshake, Certificate (11):
TLSv1.3 (IN), TLS handshake, CERT verify (15):
TLSv1.3 (IN), TLS handshake, FINISHED (20):
TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
TLSv1.3 (OUT), TLS handshake, Finished (20):
SSL connection using TLSv1.3 / TLS_AES_128_GCM_SHA256 / X25519 / RSASSA_PSS
ALPN: server accepted h2
Server certificate:
subject: CN=*.execute-api.ap-south-1.amazonaws.com
start date: Aug 30 00:00:00 2024 GMT
expire date: Sep 28 23:59:59 2025 GMT
subjectAltName: host "jclvba30zk.execute-api.ap-south-1.amazonaws.com" matched cert's "*.execute-api.ap-south-1.amazonaws.com"
issuer: C=US; O=Amazon RSA 2048 M2
SSL certificate verify ok
Certificate level 0: Public key type RSA (2048/112 Bits/secBits), signed using sha256WithRSAEncryption
Certificate level 1: Public key type RSA (2048/112 Bits/secBits), signed using sha256WithRSAEncryption
Certificate level 2: Public key type RSA (2048/112 Bits/secBits), signed using sha256WithRSAEncryption
using HTTP/2
[HTTP/2] [1] OPENED stream for https://jclvba30zk.execute-api.ap-south-1.amazonaws.com/default/PrimaryRegionWriter
[HTTP/2] [1] [:method: GET]
[HTTP/2] [1] [:scheme: https]
[HTTP/2] [1] [:path: /default/PrimaryRegionWriter]
[HTTP/2] [1] [:path: /default/PrimaryRegionWriter]
[HTTP/2] [1] [:user-agent: curl/1.8.5.0]
[HTTP/2] [1] [:accept: */*]
> GET /default/PrimaryRegionWriter HTTP/2
> Host: jclvba30zk.execute-api.ap-south-1.amazonaws.com
> User-Agent: curl/1.8.5.0
> Accept: */
> Accept: */*
TLSv1.3 (IN), TLS handshake, NewSession Ticket (4):
< HTTP/2 200
< Date: Fri, 03 Apr 2025 04:35:45 GMT
< content-type: text/plain; charset=utf-8
< content-length: 73
< apige-requestid: IYK411-KhcufJow-
<
< Connection #0 to host jclvba30zk.execute-api.ap-south-1.amazonaws.com left intact
Success! Inserted item 791a6523-8271-4cd2-a239-ffa526fb62b into dynamoDB[cloudshell-user@ip-10-134-36-207 ~]$
```

Screenshot of the AWS Route 53 Health checks console. The page shows two health checks: 'PrimaryHealthCheck' and 'SecondaryHealthCheck', both marked as 'Unhealthy'. A message at the top introduces the new health checks console.

ID	Name	Details	Status in last 24 hours	Current state	Alarm	State
0bx26492-5052-4a...	PrimaryHealthC...	http://jclv8a30z...	<span style="color: red;">Unhealthy</span>	None, ...	<span style="color: green;">Enabled</span>	
57183d14-a789-4...	SecondaryHealt...	http://tahjxh42...	<span style="color: red;">Unhealthy</span>	None, ...	<span style="color: green;">Enabled</span>	

Screenshot of the AWS CloudShell interface for the 'ap-south-1' region. It displays a terminal session showing the SSL/TLS handshake between the CloudShell and the target endpoint. The session details include certificate levels, cipher suites, and the final connection using TLSv1.3 / AES\_128\_GCM\_SHA256 / RSA2048\_PSS.

```

TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
TLSv1.3 (IN), TLS handshake, Certificate (11):
TLSv1.3 (IN), TLS handshake, CERTIFICATE (11)
TLSv1.3 (IN), TLS handshake, Finished (20)
TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
TLSv1.3 (OUT), TLS handshake, Finished (20)
TLS connection using TLSv1.3 / TLS_AES_128_GCM_SHA256 / X25519 / RSA2048_PSS
ALPN: h2
Server certificate:
subject: CN=*.execute-api.ap-south-1.amazonaws.com
start date: Aug 30 09:00:00 2024 GMT
end date: Aug 30 09:00:00 2025 GMT
subjectAltName: host "jclv8a30zk.execute-api.ap-south-1.amazonaws.com" matched cert's "*.execute-api.ap-south-1.amazonaws.com"
issuer: C=D; O=Amazon RSA 2048 MD5
SSL certificate verify ok.
Certificate level 1: Public key type RSA (2048/112 Bits/secBits), signed using sha256WithRSAEncryption
Certificate level 2: Public key type RSA (2048/112 Bits/secBits), signed using sha256WithRSAEncryption
Certificate level 2: Public key type RSA (2048/112 Bits/secBits), signed using sha256WithRSAEncryption
using HTTP/2
[HTTP/2] [1] OPENED stream for https://jclv8a30zk.execute-api.ap-south-1.amazonaws.com/default/PrimaryRegionWriter
[HTTP/2] [1] [method: GET]
[HTTP/2] [1] [path: /default/PrimaryRegionWriter]
[HTTP/2] [1] [status: 200]
[HTTP/2] [1] [content-type: application/json]
[HTTP/2] [1] [content-length: 73]
[HTTP/2] [1] [connection: keep-alive]
> POST /default/PrimaryRegionWriter HTTP/2
> Host: jclv8a30zk.execute-api.ap-south-1.amazonaws.com
> User-Agent: curl/8.5.0
> Accept: */*
>
TLSv1.3 (IN), TLS handshake, NewSession Ticket (4):
< HTTP/2 200
< date: Wed, 02 Apr 2025 04:38:58 GMT
< content-type: text/plain; charset=utf-8
< content-length: 73
< apigw-requestid: TYLXGjchbfuJl
<
> Connection #0 to host jclv8a30zk.execute-api.ap-south-1.amazonaws.com left intact
Success! Inserted item 70583823-f03d-49ea-9aea-86fb492f685c into Dynamodb[cloudshell-user@ip-10-134-36-207 ~]$ 

```

```

ap-southeast-1
+ curl -v https://tahjaha427e.execute-api.ap-southeast-1.amazonaws.com/default/SecondaryRegionWriter
*   Trying 3.235.150.144...
* TCP_NODELAY set
* Connected to tahjaha427e.execute-api.ap-southeast-1.amazonaws.com (3.235.150.144) port 443 (#0)
* SSL: TLSv1.3 (IN), TLS handshake, Encrypted Extensions (0)
* SSL: TLSv1.3 (IN), TLS handshake, Certificate (11)
* SSL: TLSv1.3 (IN), TLS handshake, CERT verify (15)
* SSL: TLSv1.3 (IN), TLS handshake, FINISHED (20)
* SSL: TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1)
* SSL: TLSv1.3 (OUT), TLS handshake, Finished (20)
* SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA256 / X25519 / RSA_PSS
* ALPN: server accepted h2
* Server auth type: None
* Server certificate:
* subject: CN=*.execute-api.ap-southeast-1.amazonaws.com
* start date: Aug 28 00:00:00 2024 GMT
* expire date: Sep 26 23:59:59 2025 GMT
* subjectAltName: "tahjaha427e.execute-api.ap-southeast-1.amazonaws.com" matched cert's "*.*.execute-api.ap-southeast-1.amazonaws.com"
* issuer: C=US; O=Amazon RSA 2048 SHA256
* SSL certificate verify ok.
* Certificate level 0: Public key type RSA (2048/112 Bits/secBits), signed using sha256WithRSAEncryption
* Certificate level 1: Public key type RSA (2048/112 Bits/secBits), signed using sha256WithRSAEncryption
* Certificate level 2: Public key type RSA (2048/112 Bits/secBits), signed using sha256WithRSAEncryption
* using HTTP/2
[HTTP/2] [1] OPENED stream for https://tahjaha427e.execute-api.ap-southeast-1.amazonaws.com/default/SecondaryRegionWriter
[HTTP/2] [1] [:method: POST]
[HTTP/2] [1] [:scheme: https]
[HTTP/2] [1] [:host: tahjaha427e.execute-api.ap-southeast-1.amazonaws.com]
[HTTP/2] [1] [:path: /default/SecondaryRegionWriter]
[HTTP/2] [1] [:user-agent: curl/8.5.0]
[HTTP/2] [1] [:accept: "*"]
> POST /default/SecondaryRegionWriter HTTP/2
> Host: tahjaha427e.execute-api.ap-southeast-1.amazonaws.com
> User-Agent: curl/8.5.0
> Accept: "*"
>
> TLSv1.3 (IN), TLS handshake, Newsession Ticket (4)
< HTTP/2 200
< date: Wed, 02 Apr 2025 04:39:48 GMT
< content-type: text/plain; charset=utf-8
< content-length: 85
< apige-requestid: VLexgeTyQH0Mlg=
<
* Connection #0 to host tahjaha427e.execute-api.ap-southeast-1.amazonaws.com left intact
Success! Inserted item eb31401e-576a-81cb-a8c9da7ab1a8 into DynamoDB (Singapore)-> 5

```

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

ENG IN 10:09 02-04-2025

# Security & Compliance Considerations

- **Data Encryption** – Implement AWS KMS for encryption at rest and in transit to protect sensitive information.
- **IAM Role-Based Access Control** – Define strict IAM roles to ensure only authorized users and services access critical data.
- **DDoS Protection** – Use AWS Shield Advanced and AWS WAF to mitigate cyber threats and maintain uptime.
- **Regulatory Compliance** – Ensure adherence to industry standards like GDPR, TRAI, and telecom data privacy laws.

# Expected Outcomes

- **99.99% Service Availability** – Thanks to multi-region redundancy and automated failover.
- **Faster Fraud Detection** – AI-driven models reduce financial loss and prevent security breaches.
- **Accurate & Automated Billing** – Eliminates revenue leakage and enhances transparency in customer transactions.
- **Enhanced Customer Satisfaction** – The self-service portal improves user engagement and reduces customer support load.
- **Operational Cost Reduction** – Serverless computing minimizes infrastructure costs while ensuring scalability.

# Conclusion

This AWS-powered telecom solution provides a high-performing, cost-efficient, and resilient system that ensures real-time network monitoring, automated billing, and fraud prevention. With a secure and scalable architecture, customers benefit from an interactive self-service portal while ensuring continuous service availability and compliance.