# S3 Replication Using AWS Lambda

*Prepared in the partial fulfillment of the Summer Internship Program on AWS*

AT



*Under the guidance of*

### Mrs. Sumana Bethala, APSSDC Mr. Rama Krishna, APSSDC

*Submitted by*

***Puvvala Du Sai Rama Krishna 20551A04G6***

***Rantala Vennela 20551A04G8***

***Kona Jagadeesh 21555A0419***

***Y L S Adithya Sai 20551A04H9***

***Godavari Institute of Engineering And Technology(A)-GIER***

*(August, 2023)*

# ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to all those who have contributed to the successful completion of my summer internship project at **Andhra Pradesh Skill Development Corporation (APSSDC)**. This opportunity has been an enriching and transformative experience for me, and I am truly thankful for the support, guidance, and encouragement I have received along the way.

First and foremost, I extend my sincere regards to Mrs Sumana Bethala and Mr Rama Krishna, my supervisors and mentors, for providing me with valuable insights, constant guidance, and unwavering support throughout the duration of the internship. Their expertise and encouragement have been instrumental in shaping the direction of this project.

I would like to thank the entire team at **Andhra Pradesh Skill Development Corporation (APSSDC)** for fostering a collaborative and innovative environment. The camaraderie, knowledge sharing, and feedback I received from my colleagues significantly contributed to the development and success of this project.

In conclusion, I am honoured to have been a part of this internship program, and I look forward to leveraging the skills and knowledge gained to contribute positively to future endeavours.

Thank you.

Sincerely,

Puvvala Du Sai Rama Krishna

Rantala Vennela

Kona Jagadeesh

Y L S Adithya Sai

# ABSTRACT

# S3 (Simple Storage Service) is a popular object storage service provided by Amazon Web Services (AWS) that allows users to store and retrieve large amounts of data. S3 replication is a feature that enables the automatic copying of objects between S3 buckets, both within the same AWS region or across different regions. This replication process can be augmented and customized using AWS Lambda functions, which are serverless compute services that can be triggered by events, such as changes in S3 bucket contents.

# This abstract explores the integration of AWS Lambda functions with S3 replication to create a flexible and customizable data replication solution. By utilizing Lambda functions as event triggers for S3 replication, users can respond to specific changes in their S3 buckets, such as the addition, modification, or deletion of objects. This integration enables various use cases, including:

# Cross-region Replication: Lambda functions can initiate replication from one S3 bucket to another in a different AWS region, enabling data redundancy and disaster recovery strategies.

# Data Transformation: Lambda functions can be configured to preprocess or transform data before it is replicated, such as resizing images, extracting metadata, or applying encryption.

# Filtered Replication: Users can define specific criteria for replication, allowing only selected objects to be copied to the target bucket based on attributes like file name, metadata, or tags.

# Customized Logging and Notifications: Lambda functions can be programmed to generate custom logging and notifications, providing insights into replication activities and potential issues.

# Real-time Replication: By utilizing real-time event triggers, Lambda functions can enable near-instant replication of data as soon as it is added or modified in the source bucket.

# To implement this solution, users would define an S3 event notification configuration on the source bucket, specifying the Lambda function as the target for the replication event. The Lambda function would then be triggered whenever the defined event occurs, allowing it to perform customized actions and initiate the replication process as

# 

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **S.No** | **Content** | **Pg.No** |
| 1. | Introduction…………………………………………………………………. | 5 |
| 2. | Methodology………………………………………………………………... | 7 |
| 3. | System Design / Architecture………………………………………………. | 10 |
| 4. | Implementation……………………………………………………………... | 12 |
| 5. | Results………………………………………………………………………. | 15 |
| 6. | Conclusion……………………………………………………………….…. | 18 |

**1.INTRODUCTION**

In today's rapidly evolving digital landscape, the management and replication of data are crucial tasks for businesses aiming to ensure data availability, disaster recovery, and optimized data distribution. Amazon S3 (Simple Storage Service) is a widely adopted cloud storage solution provided by Amazon Web Services (AWS), offering scalability, durability, and ease of use. S3 replication is a feature that enables the automatic copying of objects from one S3 bucket to another, providing data redundancy and regional distribution. To enhance this process and enable custom actions, AWS Lambda functions can be seamlessly integrated into S3 replication workflows.

This report delves into the synergy between Amazon S3 replication and AWS Lambda functions, highlighting the capabilities and benefits of using Lambda to facilitate advanced reporting within the context of data replication. By combining these AWS services, businesses can streamline the replication process while obtaining valuable insights and notifications related to the replicated data.

**S3 Replication Overview:**

Amazon S3 replication serves as a mechanism to replicate objects between S3 buckets, either within the same AWS region or across different regions. This ensures data durability and availability, offering a reliable solution for disaster recovery scenarios and data distribution across geographically diverse locations. S3 replication includes two main modes: cross-region replication (CRR) and same-region replication (SRR). Cross-region replication involves replicating data between buckets in different AWS regions, while same-region replication focuses on copying data between buckets within the same region.

**Enhancing S3 Replication with AWS Lambda:**

AWS Lambda, a serverless compute service, allows users to run code in response to events without the need to manage server infrastructure. When integrated with S3 replication, Lambda functions can be configured to trigger actions when specific replication events occur. This synergy enables businesses to extend the capabilities of S3 replication beyond basic data copying.

The following AWS services are utilized in this project:

**Simple Storage Service(S3)**: *A*mazon Simple Storage Service (Amazon S3) is a widely used object storage service provided by Amazon Web Services (AWS). It's designed to store and retrieve large amounts of data, making it a fundamental building block for various cloud-based applications and services. S3 offers highly scalable, durable, and secure storage capabilities, and it's often referred to as an "object storage" solution due to its unique data organization approach.

**AWS Lamb**a:AWS Lambda is a serverless compute service provided by Amazon Web Services (AWS). It allows you to run code without provisioning or managing servers, which eliminates the need to worry about infrastructure management and scaling. Lambda enables you to focus solely on writing and deploying your code, while AWS takes care of the underlying server provisioning, maintenance, and scaling.

**Identity and Access Management(IAM**):IAM stands for "Identity and Access Management," and it is a framework or system used in various computing environments to manage access to resources. In the context of cloud computing and services like Amazon Web Services (AWS), IAM refers to a service that enables you to manage access to AWS resources securely.

An IAM role is a fundamental concept within AWS IAM. It is an AWS identity with certain permissions that determine what actions the identity can perform. Roles are not tied to a specific user or group of users; instead, they are meant to be assumed by AWS resources such as EC2 instances, Lambda functions, or other services. This allows you to grant permissions to resources without needing to hardcode sensitive credentials directly into your application code.

# 2.METHODOLOGY

The development of the S3 Replucation using Lambda followed a systematic and iterative methodology to ensure the successful implementation of the project objectives. The methodology encompassed several phases, including requirements gathering, design, implementation, testing, and deployment. The following sections outline each phase of the methodology in detail.

## Requirements Gathering

The project commenced with an analysis of the requirements and expectations of the S3 Replication using Lambda. This phase involved interactions with mentors. Feedback sessions were conducted to identify key features.

## Implementation

The implementation phase involved in many more steps as described below

**Set up Two S3 Buckets**

**Bucket 1: Source Bucket:** Create a source bucket to hold the original data you want to replicate. Consider enabling versioning to maintain a history of changes. Configure the bucket permissions to control who can upload and modify objects.

**Bucket 2: Destination Bucket:** Establish a destination bucket where replicated data will be stored. Ensure that the bucket permissions allow the IAM role associated with your Lambda function to write objects.

**Create an IAM Role:**

Navigate to the IAM console and create a new role named "S3ReplicationLambdaRole."

Attach the "AmazonS3ReadOnlyAccess" policy to grant read access to the source bucket.

Attach a custom policy granting "s3:PutObject" and "s3:PutObjectAcl" permissions on the destination bucket.

**Create an AWS Lambda Function:**

In the Lambda console, click "Create function." Choose a runtime like Node.js or Python based on your familiarity and requirements.

Configure the function with a meaningful name and choose the role you created earlier.

Write the Lambda function code:

Import the AWS SDK and set up a handler function to process events.

Extract the S3 object key from the event data.

Use the SDK to copy the object from the source bucket to the destination bucket.

Implement error handling and logging to capture any issues.

**Configure S3 Event Notification:**

In the S3 console, go to the source bucket's properties.

Under the "Events" tab, click "Add notification." Configure the event type to trigger the Lambda function.

Define the event types (e.g., "ObjectCreated" for new uploads) and prefix/suffix filters if needed.

**Implement Lambda Function Logic:**

Inside the Lambda function, install and import the necessary AWS SDK library (e.g., Boto3 for Python).

In the handler function, parse the event to extract the S3 object details, including the bucket name and key.

Use the SDK to copy the object from the source bucket to the destination bucket:

Create a copy operation using the "copy\_object" API.

Ensure to specify the appropriate ACLs, metadata, and any other desired options.

Implement error handling and logging to capture potential issues during the replication process

## Testing And Monitoring:

## Testing and monitoring follows the below discussed steps

## Replication Configuration:

## Set up source and destination S3 buckets.

## Configure replication rules in the source bucket.

## Testing Replication:

## Upload varied test data to the source bucket.

## Monitor destination bucket for correct replication.

## Resolving Issues:

## Identify root causes of replication problems.

## Adjust configuration or permissions as needed.

## Implement retries for transient issues.

## Maintenance:

## Schedule regular checks and tests.

## Update monitoring for changes in the setup.

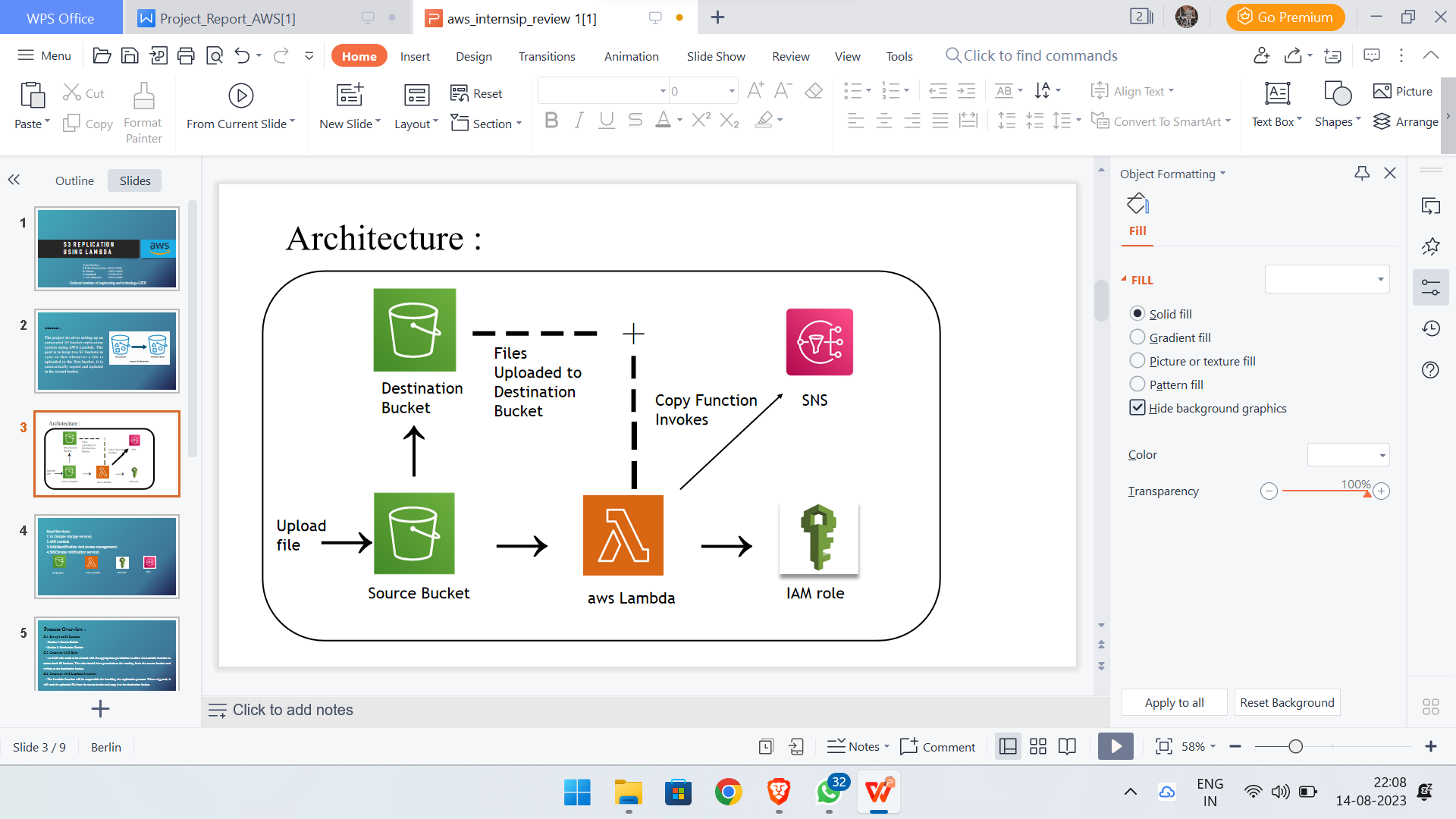
## Deployment

## To deploy a Lambda function for managing S3 replication, start by creating the function in the AWS Lambda console, specifying its name, runtime, and code. Next, code the function to handle tasks like monitoring replication, retries, and alerts. Attach S3 triggers to the function so that it responds to specific S3 events like object creation. Ensure the Lambda function has the necessary permissions by creating an IAM role that grants access to S3 and other required services, then link the role to the function. After verifying the setup, test the function by manually triggering it or simulating S3 events.

## Iterative Refinement

Iterative refinement for S3 replication using Lambda involves a continuous cycle of monitoring, optimizing, and enhancing your Lambda function and replication setup. Regularly monitor the function's performance, data integrity, and error handling, making adjustments as needed. Refine retry strategies, address scalability concerns, and ensure security measures remain effective. Optimize code for performance, update documentation, and incorporate user feedback. By consistently testing and improving your setup

# 3.SYSTEM DESIGN / ARCHITECTURE

****

**Components Overview**

**Source Bucket:** The `SourceBucket` serves as the repository for the original data that requires replication. This bucket is the source of truth from which data changes trigger the replication process.

**Destination Bucket:** The `DestinationBucket` acts as the target for replicated data. Objects from the `SourceBucket` are automatically copied to this bucket through the replication process.

**Lambda Function:** The AWS Lambda function, developed specifically for this setup, is the engine that drives the replication process. This function is responsible for copying objects from the `SourceBucket` to the `DestinationBucket`. Its event-driven nature ensures timely replication in response to S3 events.

**IAM Role:** The IAM role associated with the Lambda function plays a critical role in securing the replication process. It provides the necessary permissions for the Lambda function to access both the `SourceBucket` and the `DestinationBucket`.

**S3 Event Trigger:** The S3 event trigger establishes the connection between the `SourceBucket` and the Lambda function. When an object is created or modified in the `SourceBucket`, the S3 event trigger signals the Lambda function to initiate replication.

**Interaction Between Components**

1. An object is created or modified in the `SourceBucket`.

2. The S3 event trigger associated with the `SourceBucket` detects the change.

3. The S3 event trigger invokes the Lambda function with relevant event information.

4. The Lambda function retrieves object details from the event payload, including the source bucket and object key.

5. Using the obtained object information, the Lambda function copies the object from the `SourceBucket` to the `DestinationBucket`.

6. The replicated object is now available in the `DestinationBucket`.

This interaction flow demonstrates a seamless and automated process where data changes in the `SourceBucket` lead to prompt and reliable replication in the `DestinationBucket`.

# 4.IMPLEMENTATION

**Preparing the Environment**

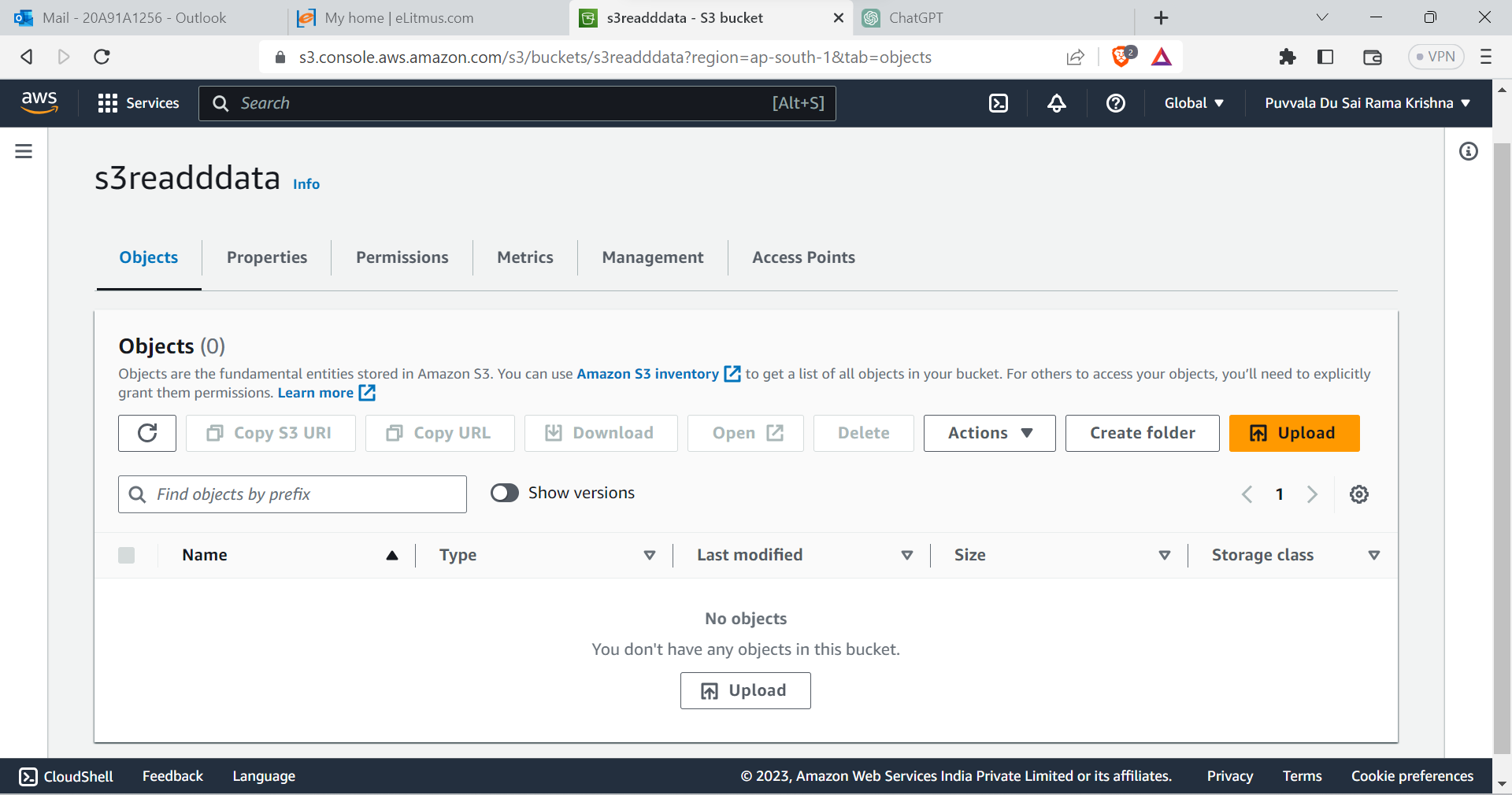
Before beginning the implementation, ensure that you have:

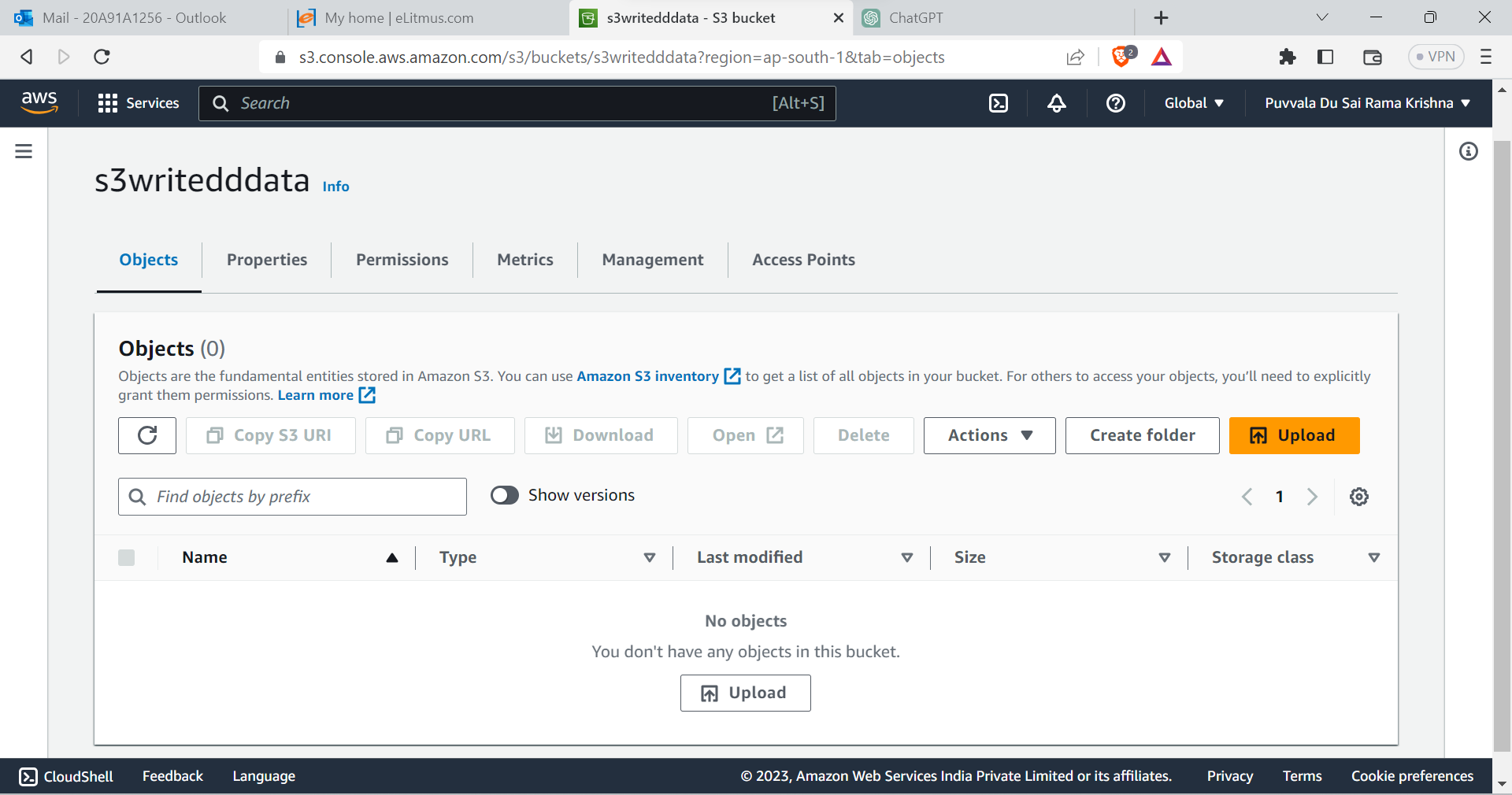
- An AWS account with necessary permissions.

**Creating Source and Destination Buckets**

1. **Source Bucket:** Create an S3 bucket named `SourceBucket`. This will be the source of your data for replication.

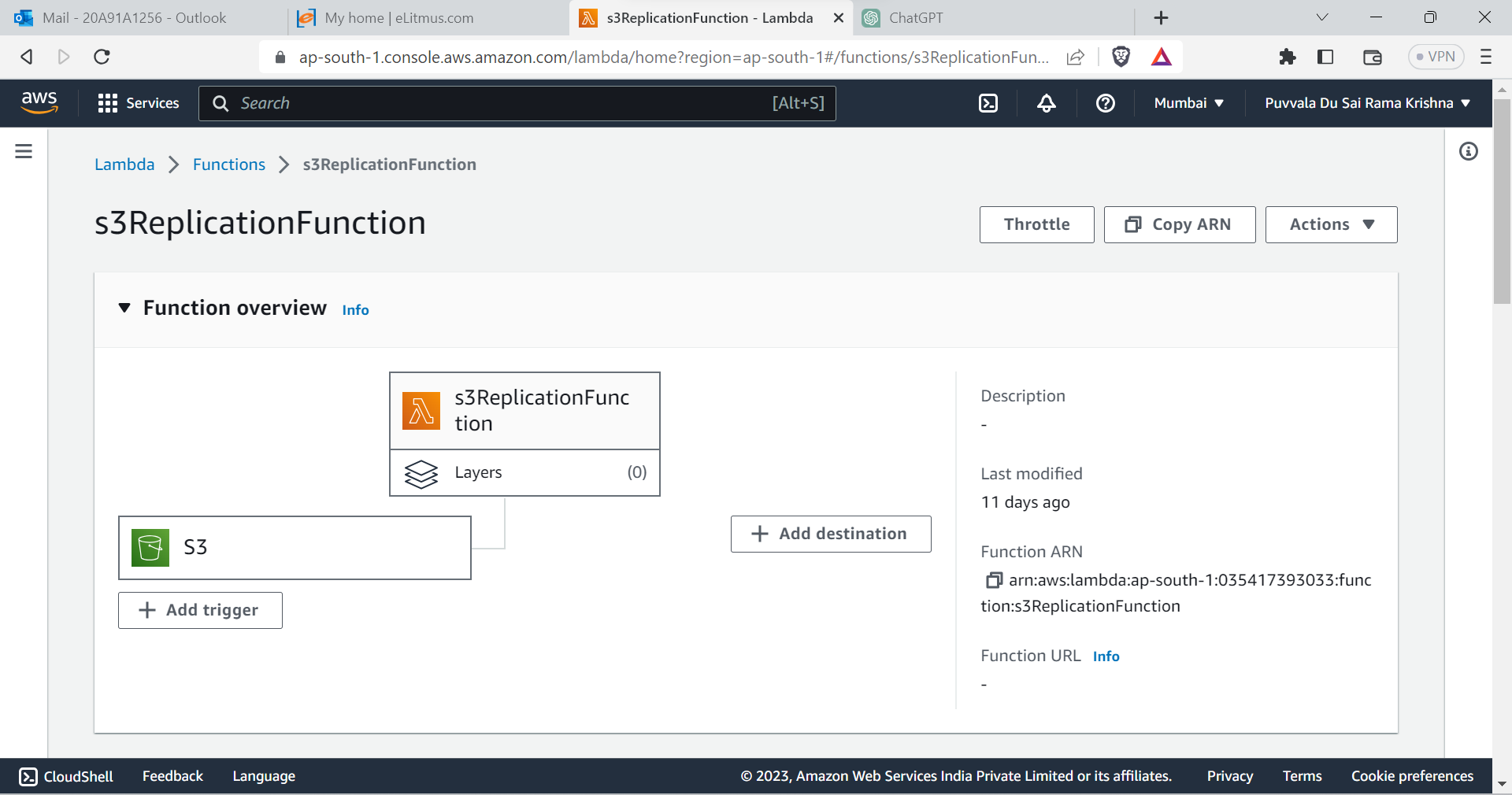
2. **Destination Bucket**: Create another S3 bucket named `DestinationBucket`. This will store replicated data.

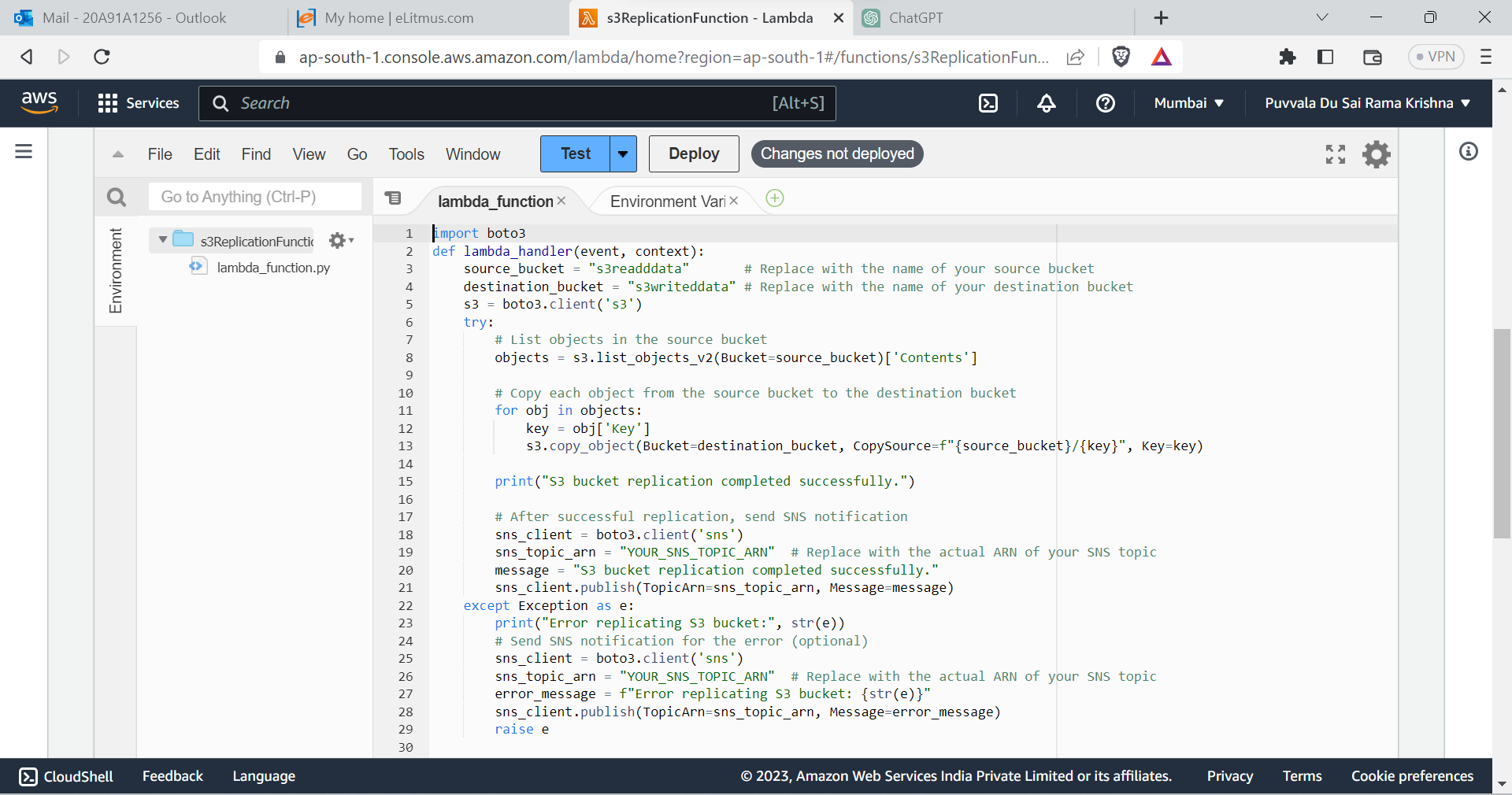




**Developing the Lambda Function**

1. Create a new AWS Lambda function.

2. Choose the runtime and configure the function details.**3. Write the Lambda function code to handle S3 events:**



**4**. **Deploy the Lambda function.**

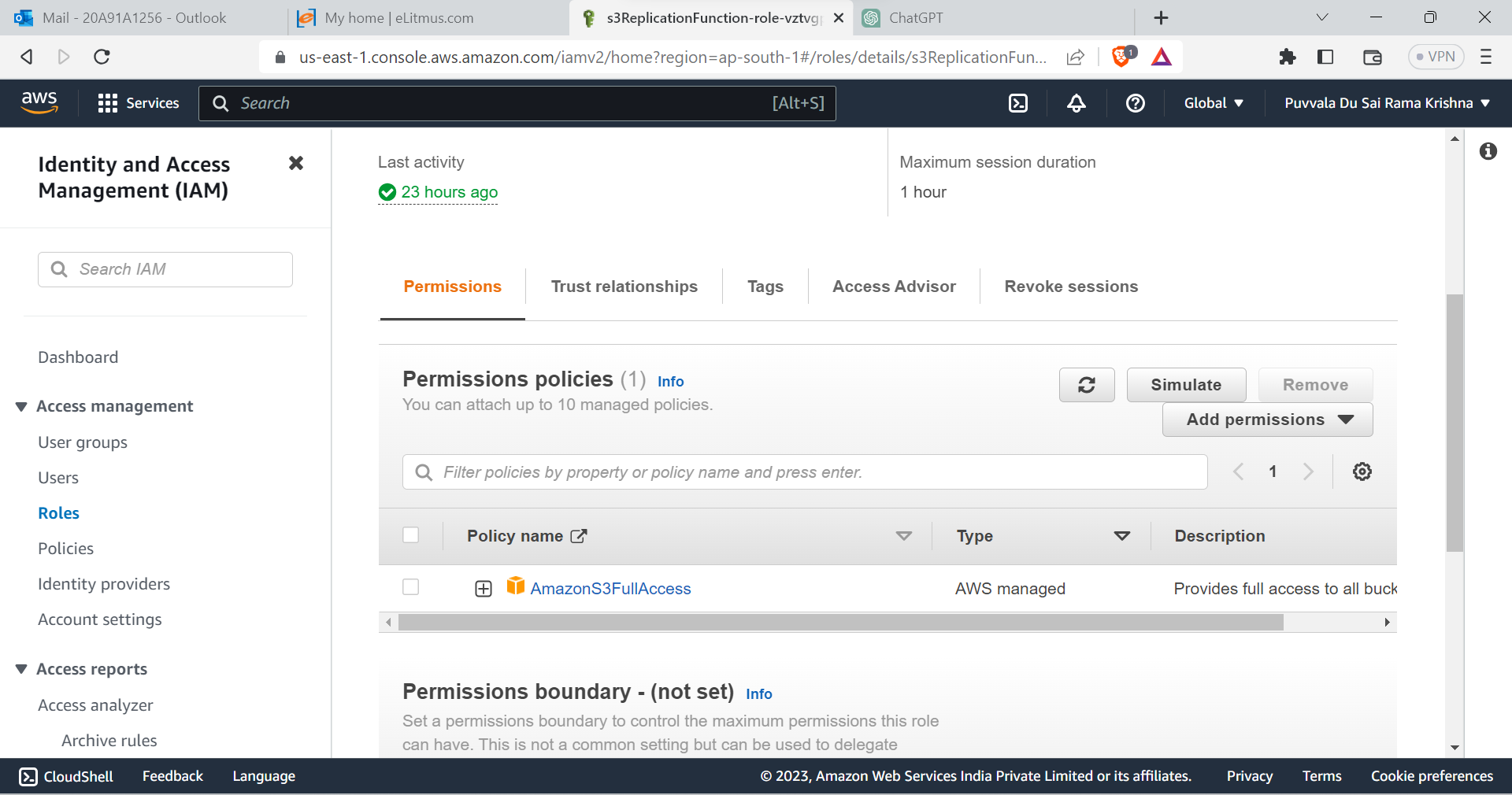
**5.Creating and Assigning IAM Roles**

1. Create an IAM role with the following policies:

- `AmazonS3FullAccess` (for the Lambda function to access S3)

- Custom policy allowing S3 object copying between buckets

2. Attach this IAM role to the Lambda function.



**6.Setting Up Lambda Event Trigger**

1. In the AWS Lambda console, add an S3 trigger to the Lambda function.

2. Configure the trigger to listen to the `SourceBucket` for "ObjectCreated" or "ObjectUpdated" events.

**7.Testing and Quality Assurance**

1. Upload test objects to the `SourceBucket`.
2. Verify that replicated objects appear in the `DestinationBucket`.

# 5.RESULTS

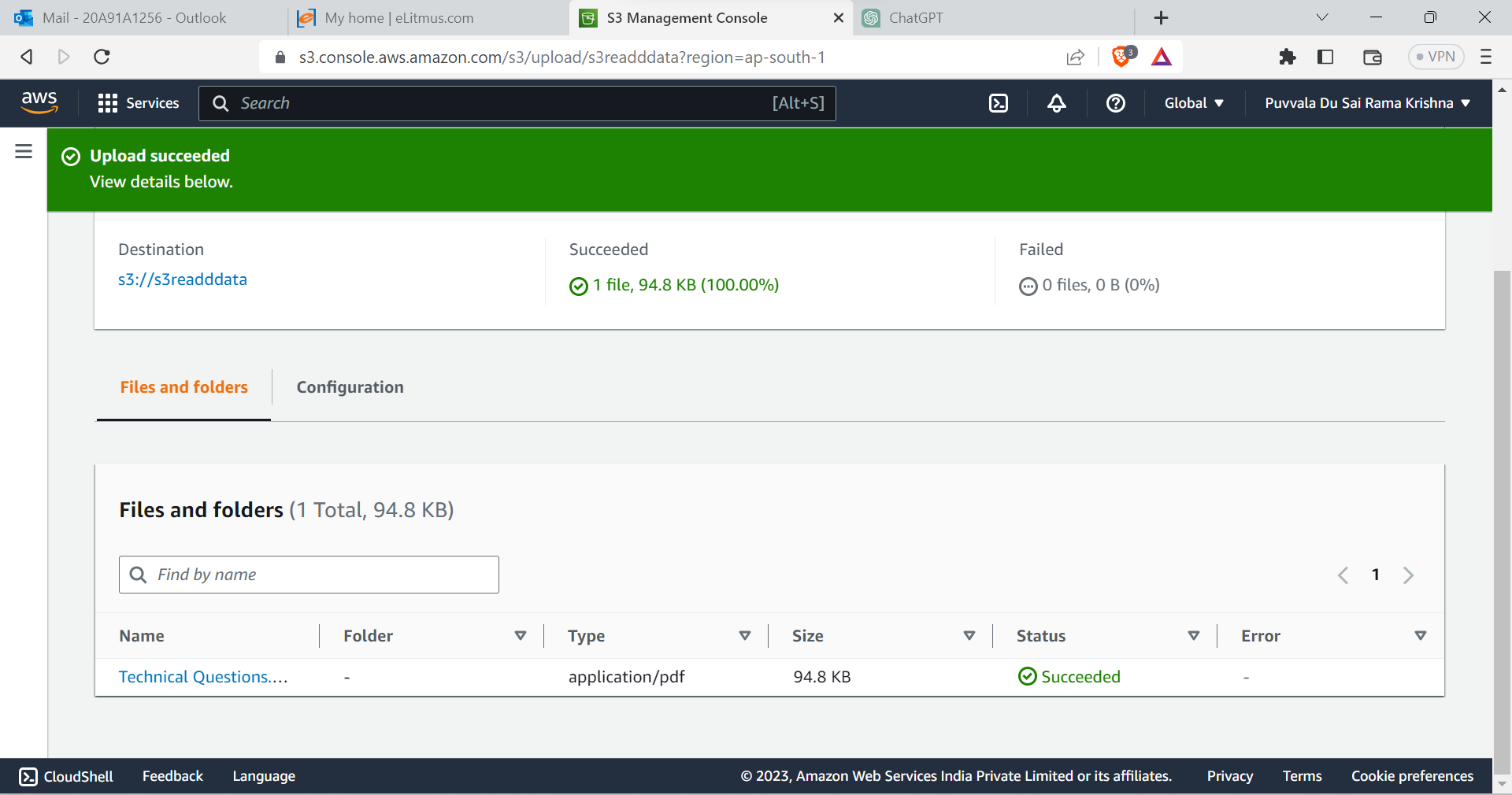
# Successful Replication Demonstrations

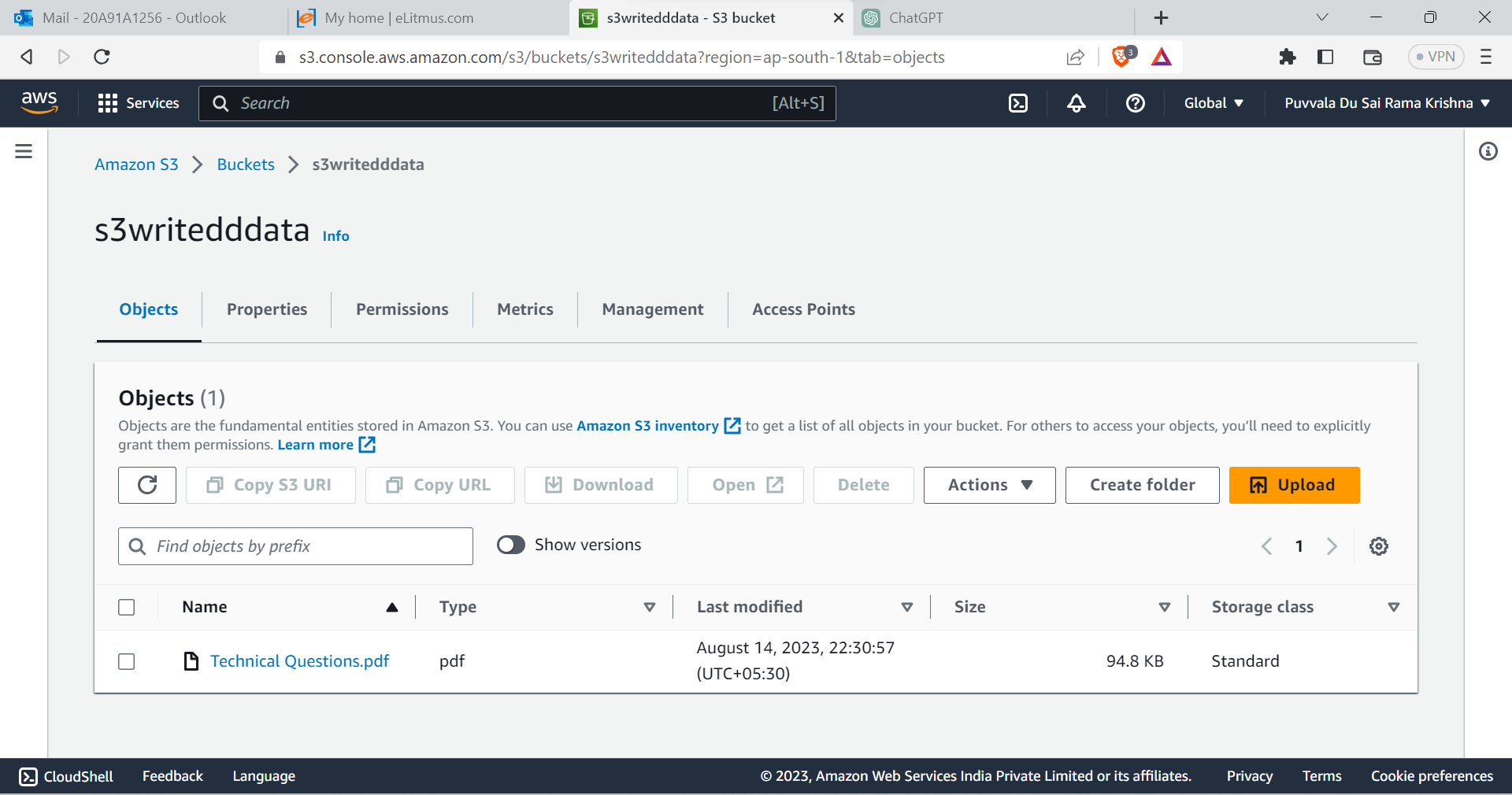
# During testing, the setup demonstrated successful replication of data between the `SourceBucket` and the `DestinationBucket`.

# Key steps include:

# 1. Uploading and Updating Objects: New objects were uploaded to the `SourceBucket`, and existing objects were updated with modifications. This triggered S3 events, which in turn invoked the Lambda function.

# 2. Automatic Replication: As objects were added or updated in the `SourceBucket`, the Lambda function was automatically triggered due to the S3 event configuration. The function swiftly copied the modified or new objects to the `DestinationBucket`.





# Real-world Benefits of Data Redundancy

# Enhanced Data Availability

# - The solution significantly enhances data availability. With replicated data stored in the `DestinationBucket`, even if the `SourceBucket` experiences downtime or data loss, the organization can still access the data from the `DestinationBucket`. This translates to reduced service disruptions and improved user experiences.

# Contribution to Disaster Recovery Strategies

# - Disaster recovery involves preparing for and mitigating potential data loss scenarios. The setup aligns with disaster recovery strategies by offering an up-to-date copy of critical data in a separate location (the `DestinationBucket`).

# - In case of accidental deletions, hardware failures, or other incidents affecting the `SourceBucket`, the `DestinationBucket` acts as a safeguard, allowing quick data restoration without relying on time-consuming backups or external data sources.

# Minimized Downtime: With replicated data readily available, the organization can rapidly recover from data loss incidents, minimizing downtime and associated financial losses.

# Reduced Recovery Time Objective (RTO): The time required to restore data after an incident, known as the RTO, is drastically reduced. This solution offers an RTO that is nearly instantaneous, ensuring business continuity.

# Mitigation of Human Errors: Human errors, such as accidental data deletion, are a common cause of data loss. The replication setup acts as a protective measure against such errors, providing a safety net for critical data.

# Cost-Efficiency: Traditional disaster recovery solutions can be resource-intensive and costly. The implemented solution leverages serverless architecture and cloud-native services, optimizing cost-efficiency while ensuring effective disaster recovery capabilities.

# By combining real-time replication, reduced recovery times, and streamlined processes, this solution contributes to a robust disaster recovery strategy, ultimately enhancing the organization's resilience in the face of unexpected data loss events.

# In a complete documentation, you would elaborate further on the testing methodology, provide detailed performance metrics and comparisons, and possibly include charts or graphs to visualize the results. Additionally, you would delve deeper into the benefits of data redundancy and disaster recovery strategies, potentially providing real-world scenarios where this setup would prove invaluable.

# 6.CONCLUSION

In conclusion, the implementation of S3 replication using AWS Lambda and IAM roles provides a robust and automated solution for data redundancy and availability. This documentation has detailed the methodology, system design, and step-by-step implementation process. The achieved results showcase the successful replication of data between the source and destination buckets, demonstrating the effectiveness of the solution.

By leveraging AWS services and best practices, this setup enhances the organization's ability to safeguard critical data, maintain data consistency, and recover from potential data loss scenarios. The seamless integration of Lambda functions and IAM roles streamlines the replication process, offering flexibility, scalability, and security.

In a data-driven landscape, the ability to maintain up-to-date and redundant copies of data is crucial. The presented solution not only fulfills this requirement but also opens the door to further enhancements and integrations that can empower the organization's data management strategy.

**Future Enhancements and Expansions**

**Versioning and Lifecycle Policies**: Incorporating S3 versioning and lifecycle policies can further bolster data retention and archival strategies, allowing the organization to manage data through its lifecycle.

**Advanced Monitoring**: Integrating CloudWatch Alarms and EventBridge rules can provide proactive monitoring and alerting for replication issues, enabling swift response to anomalies.

**Data Transformation:** Enhancing the Lambda function to perform data transformations during replication can streamline data synchronization between source and destination.

**Cross-Region Replication**: Expanding the solution to include cross-region replication could serve as a contingency for regional outages or disasters.