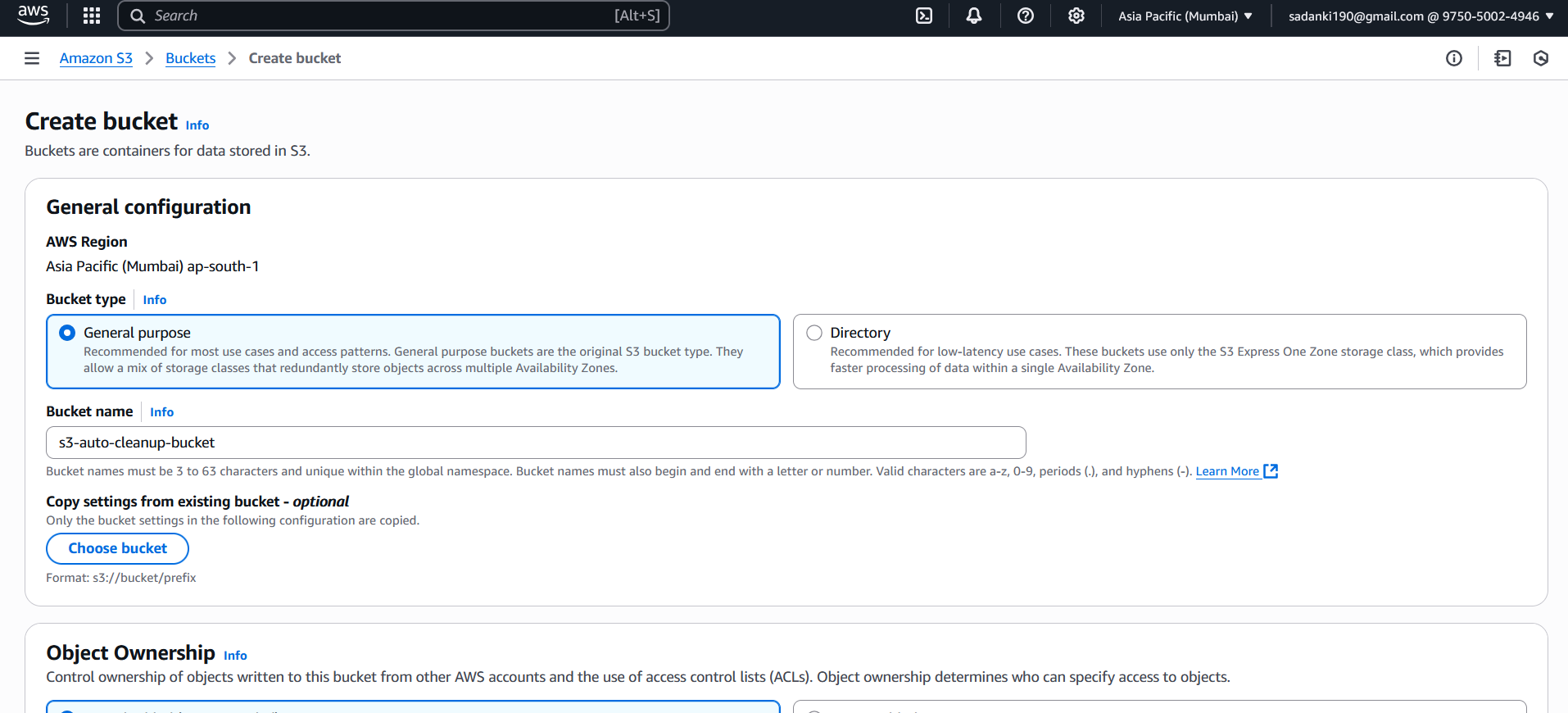
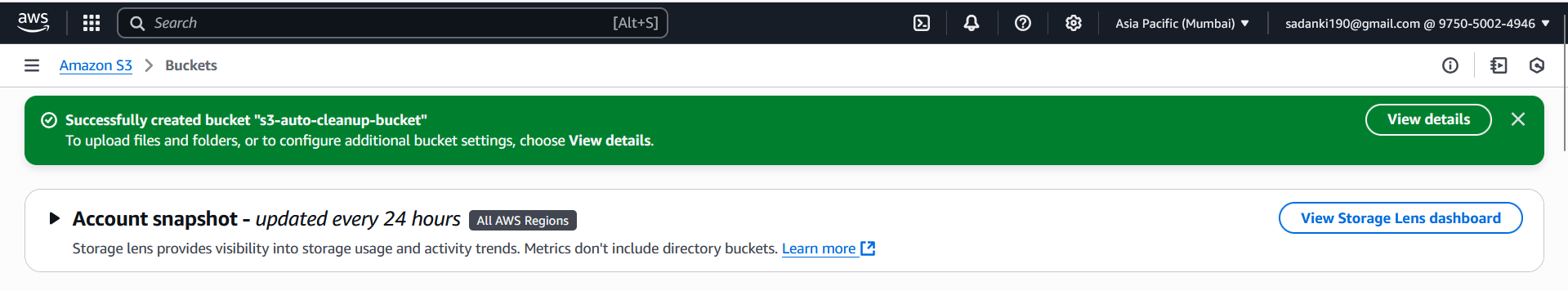
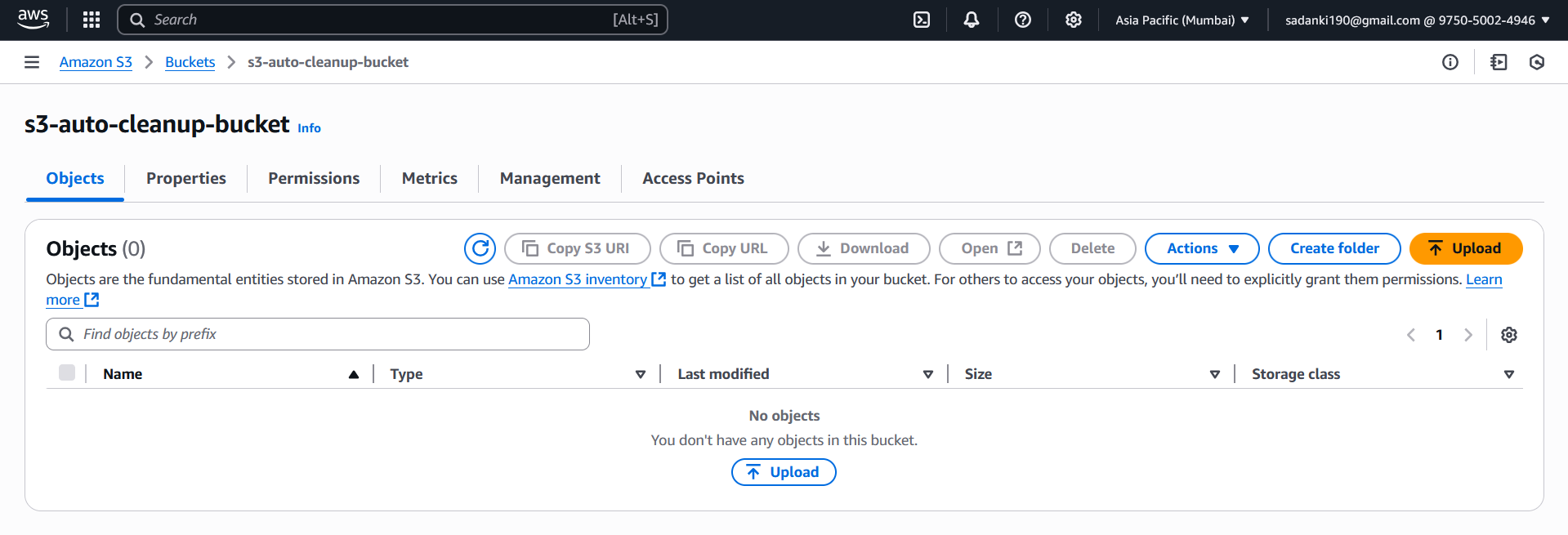
**S3 Setup:**

**Step 1: Create an S3 Bucket**

1. Go to the **AWS Management Console** → **S3**.
2. Click on **Create bucket**.
3. Enter a unique **Bucket name** (e.g., my-cleanup-bucket).
4. Select the region (same region you’ll use for Lambda, e.g., ap-south-1).
5. Leave all other settings as default and click **Create bucket**.

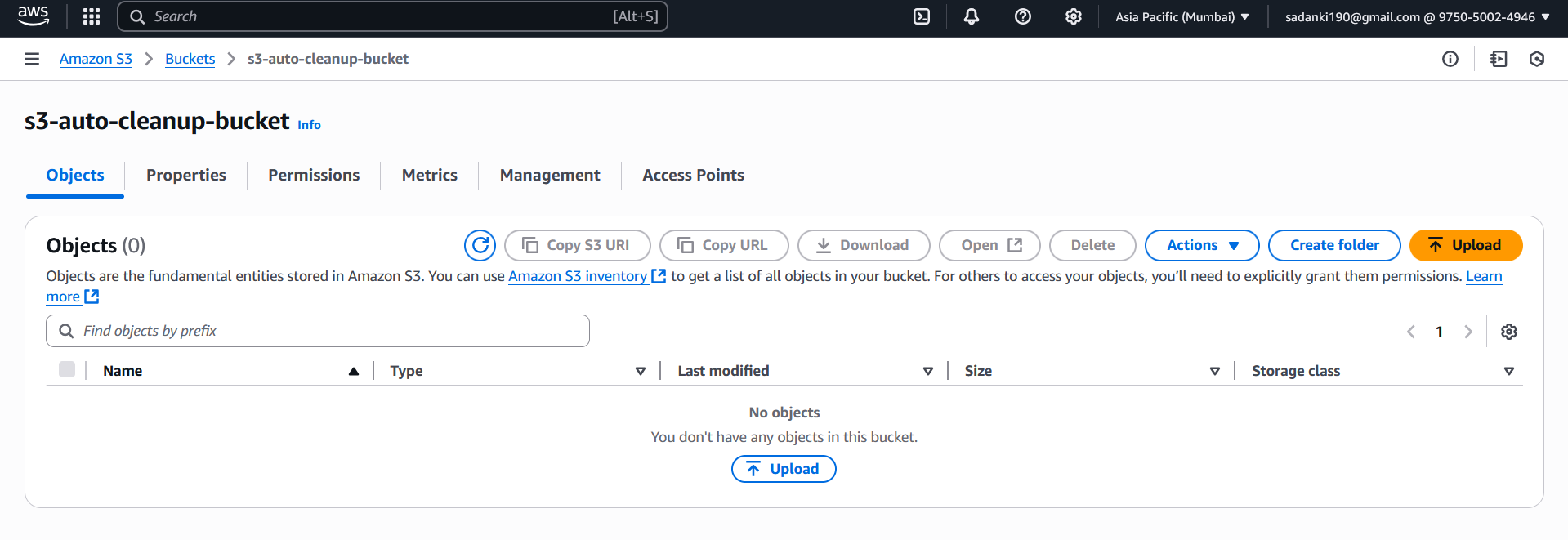


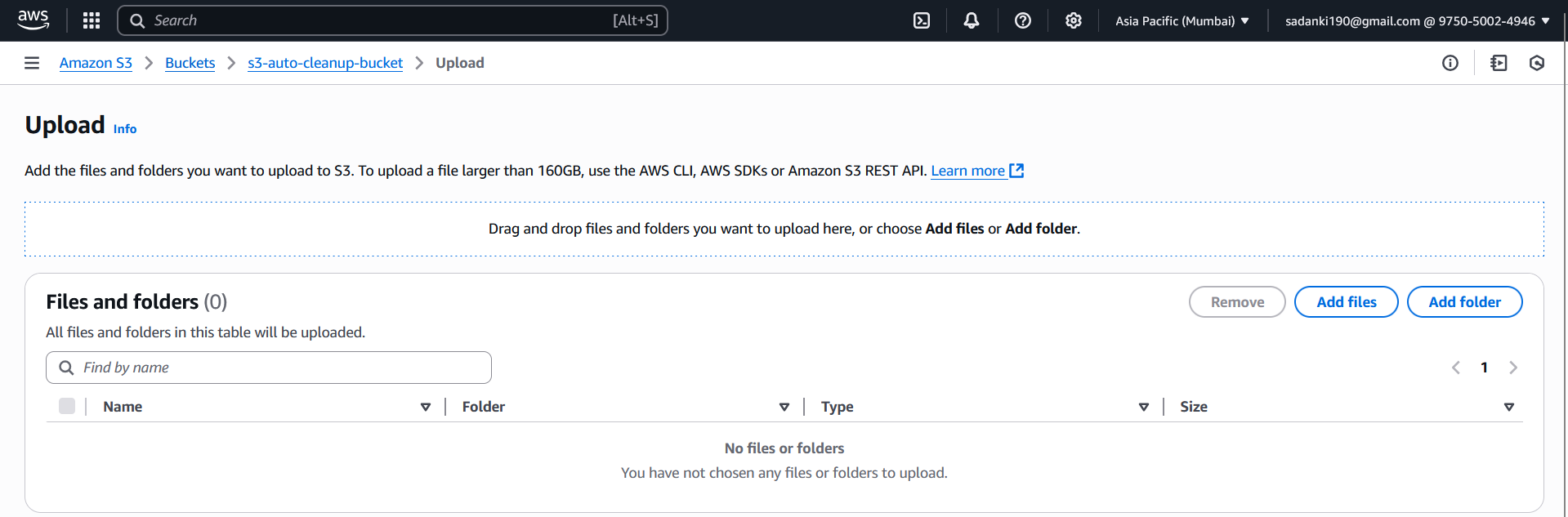


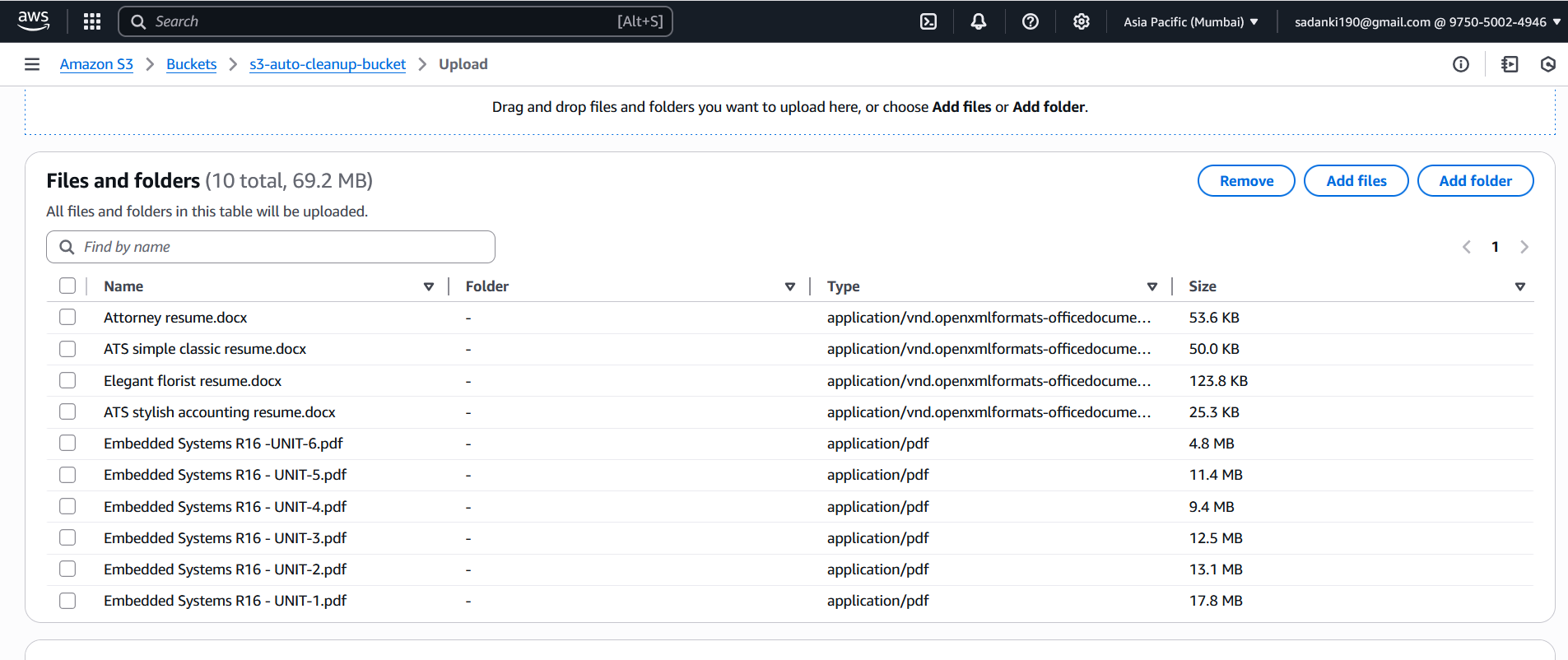


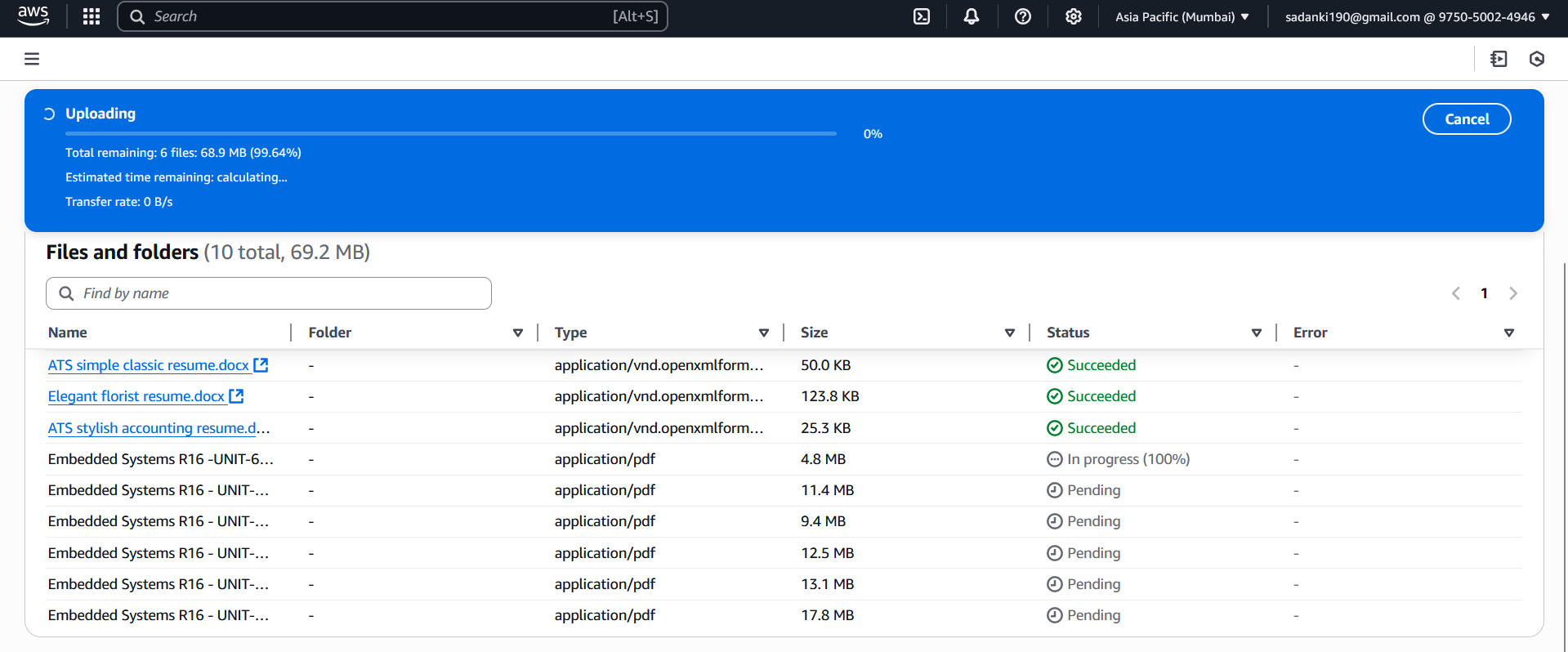
**Step 2: Upload Files to the Bucket**

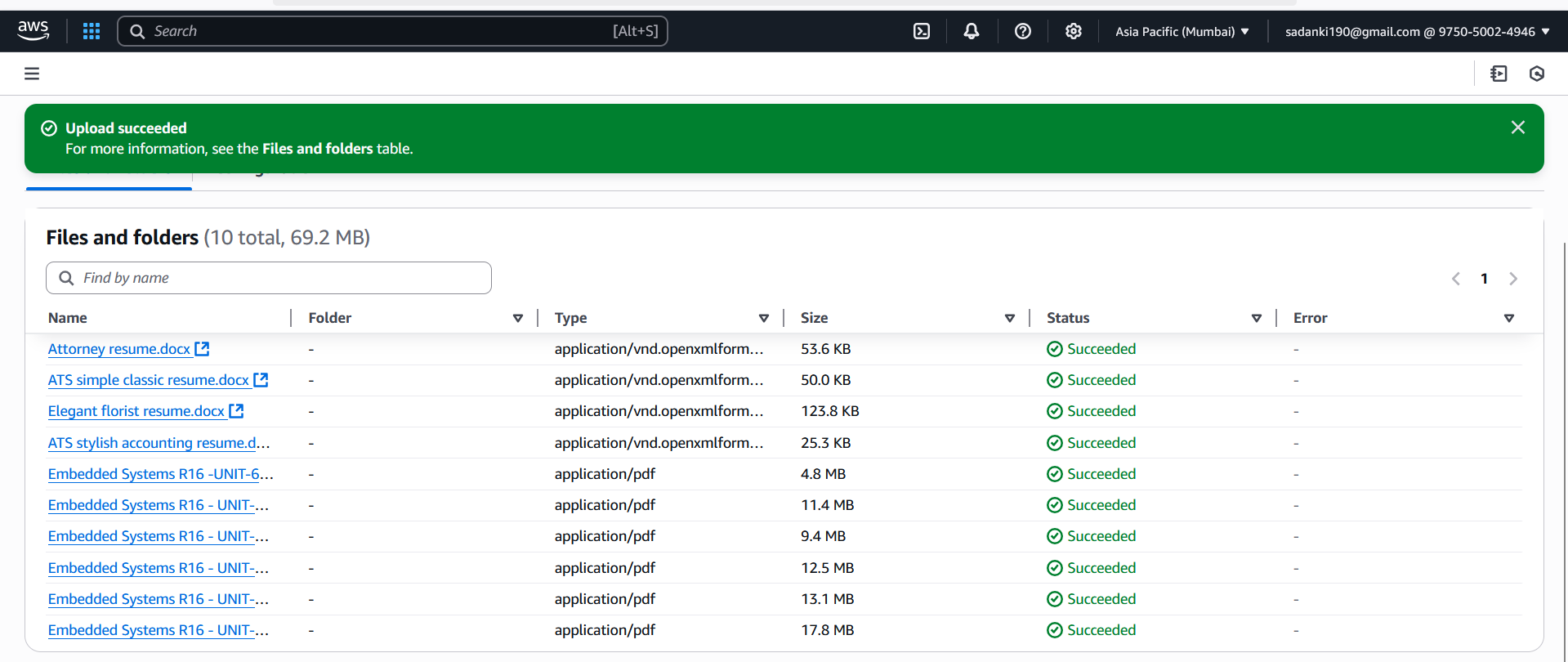
1. Open your **S3 bucket**.
2. Click **Upload** → **Add files**.
3. Select several files from your local system.
4. To simulate older files:
   * Use already old files (more than 30 days old), OR
   * Just test with recent files (you’ll see they’re not deleted).
5. Complete the upload.





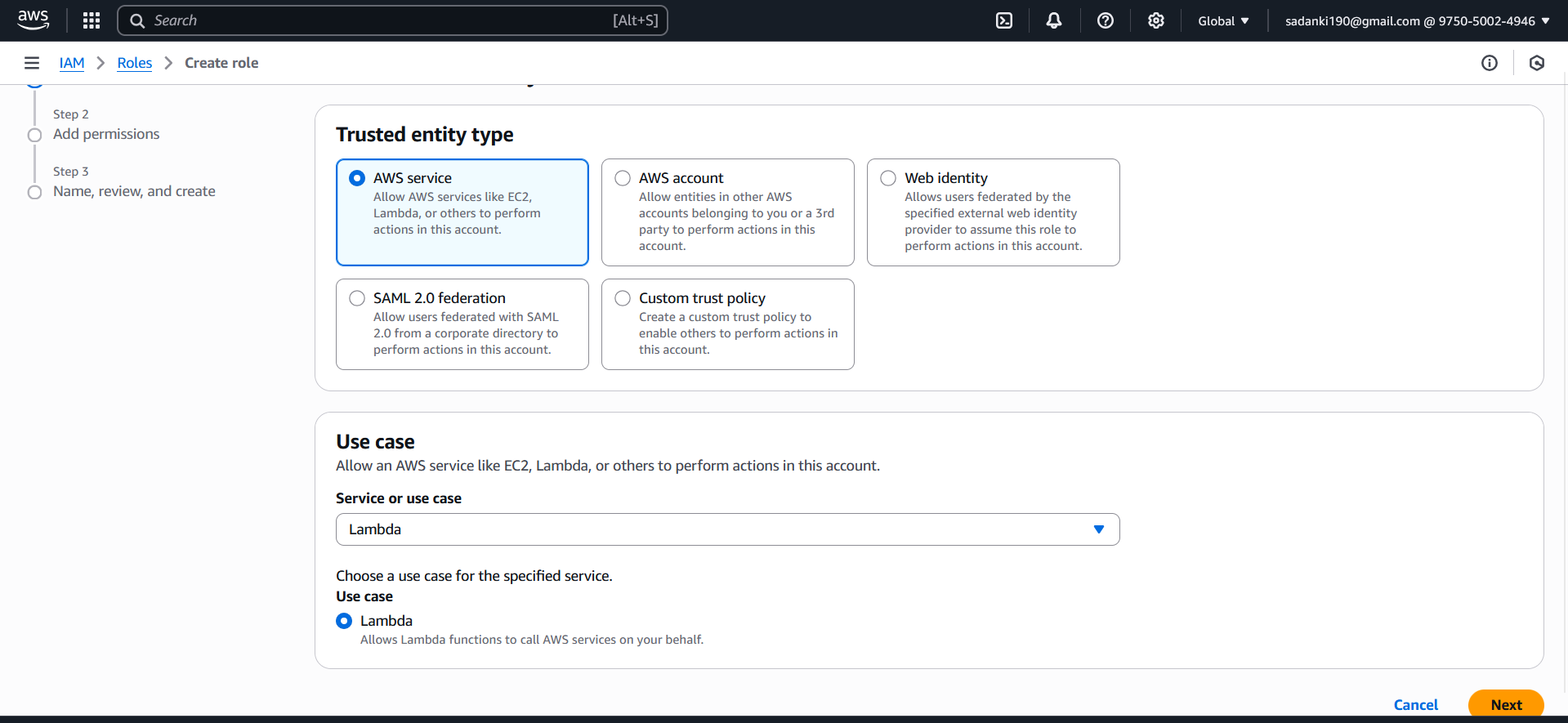


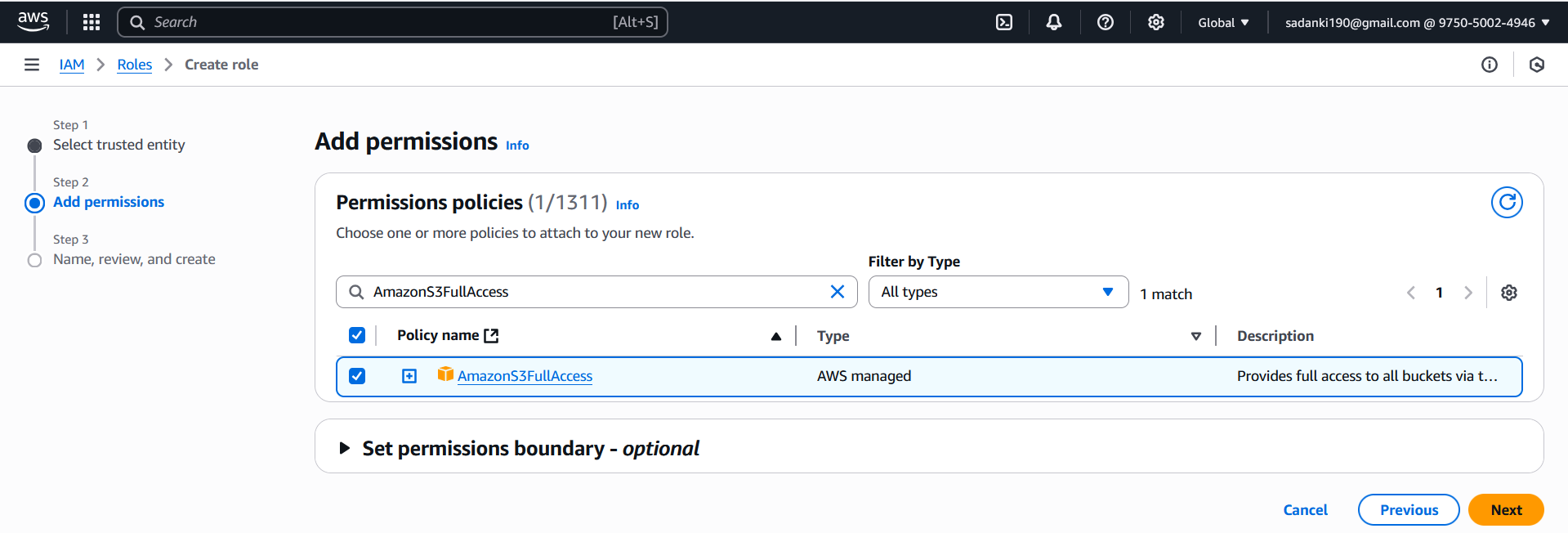


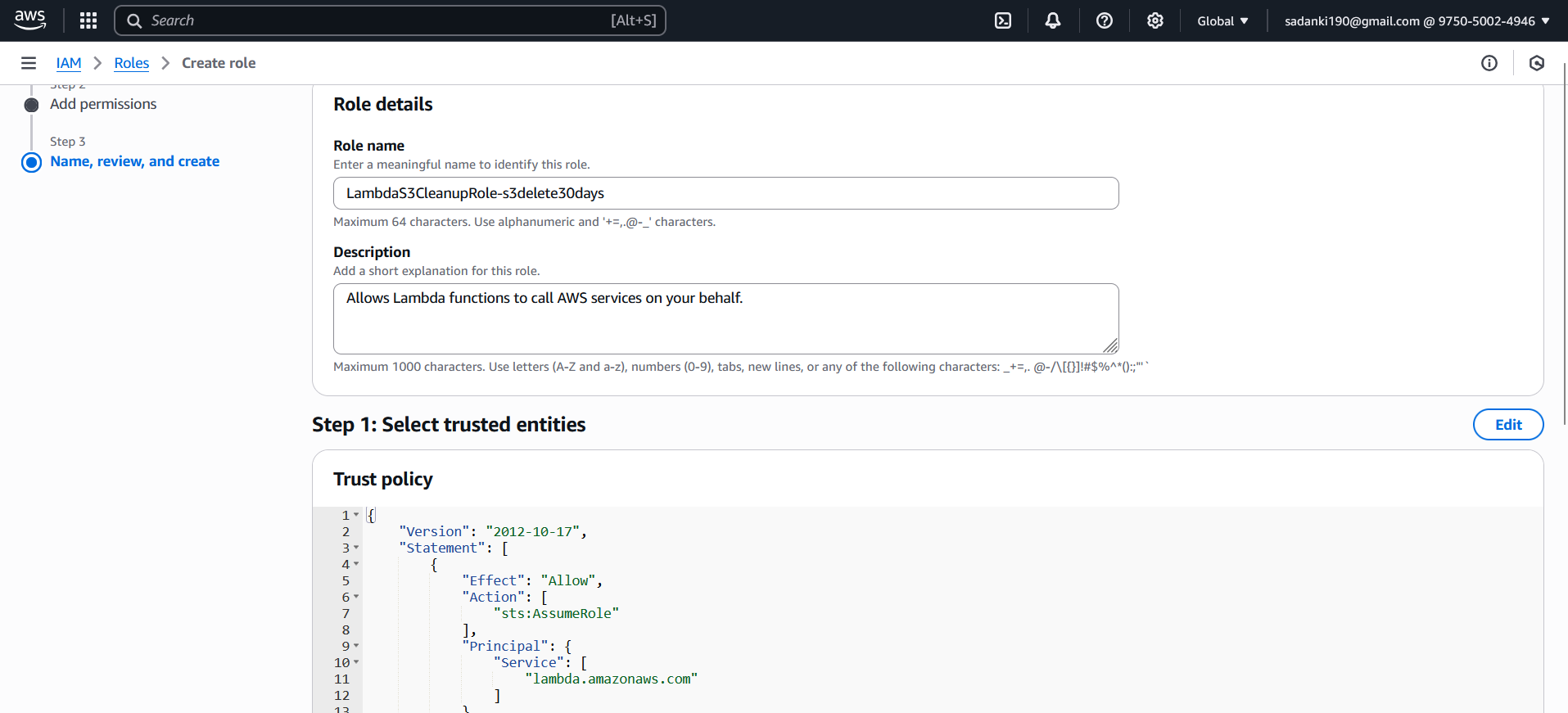


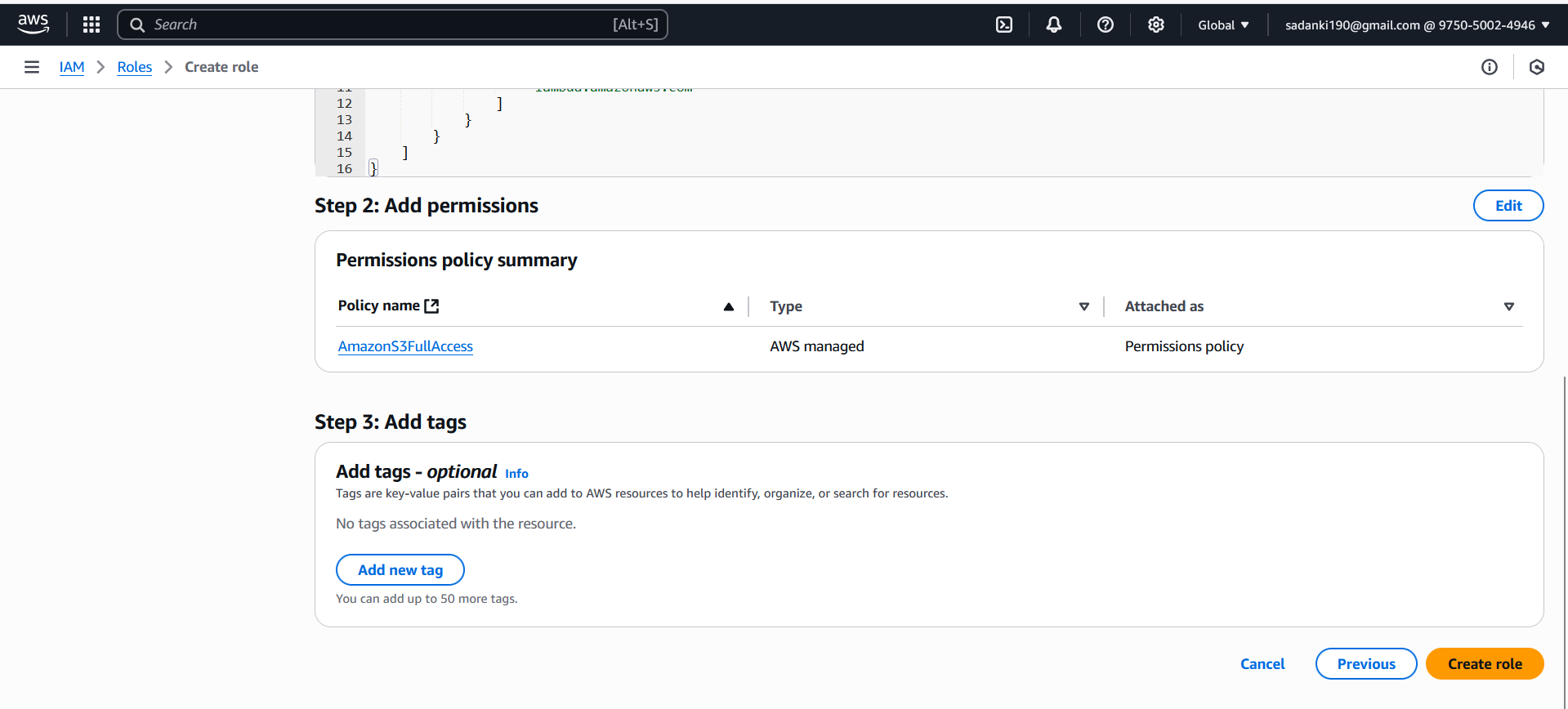
**Step 3: Create an IAM Role for Lambda**

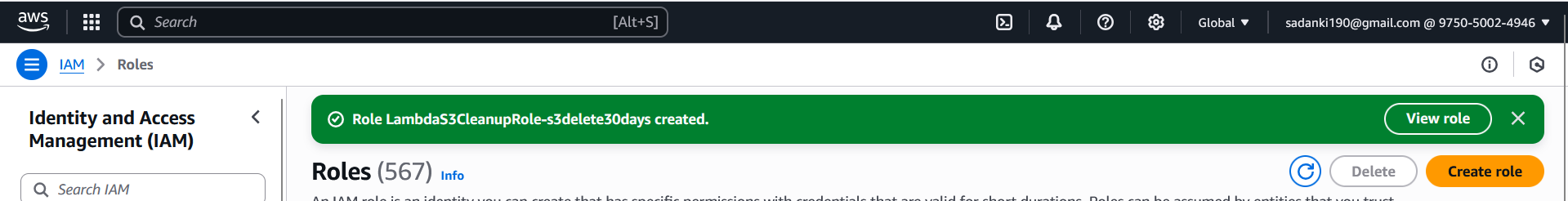
1. Go to **IAM → Roles**.
2. Click **Create role**.
3. **Select trusted entity type**: choose **AWS service**.
4. **Use case**: select **Lambda**.
5. Click **Next**.
6. **Attach permissions**: Search and select AmazonS3FullAccess.
7. Click **Next**, then give your role a name like LambdaS3CleanupRole-s3delete30days.
8. Click **Create role**.





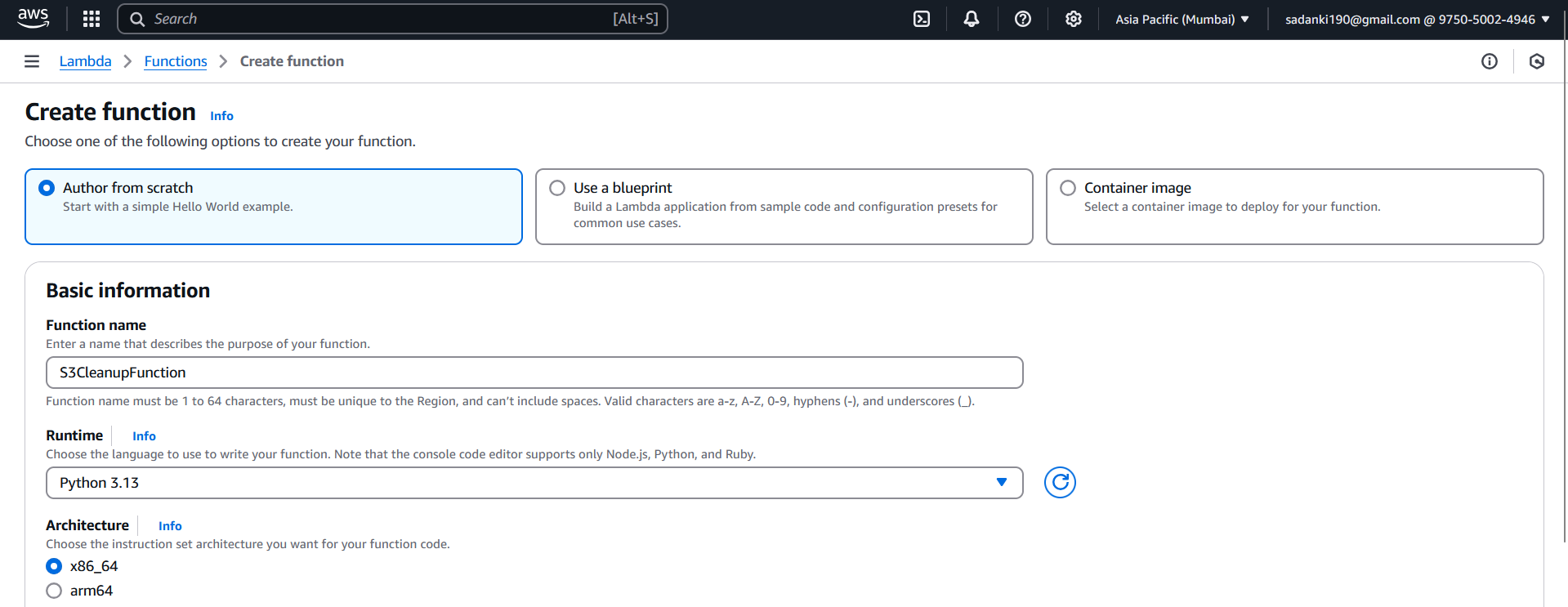


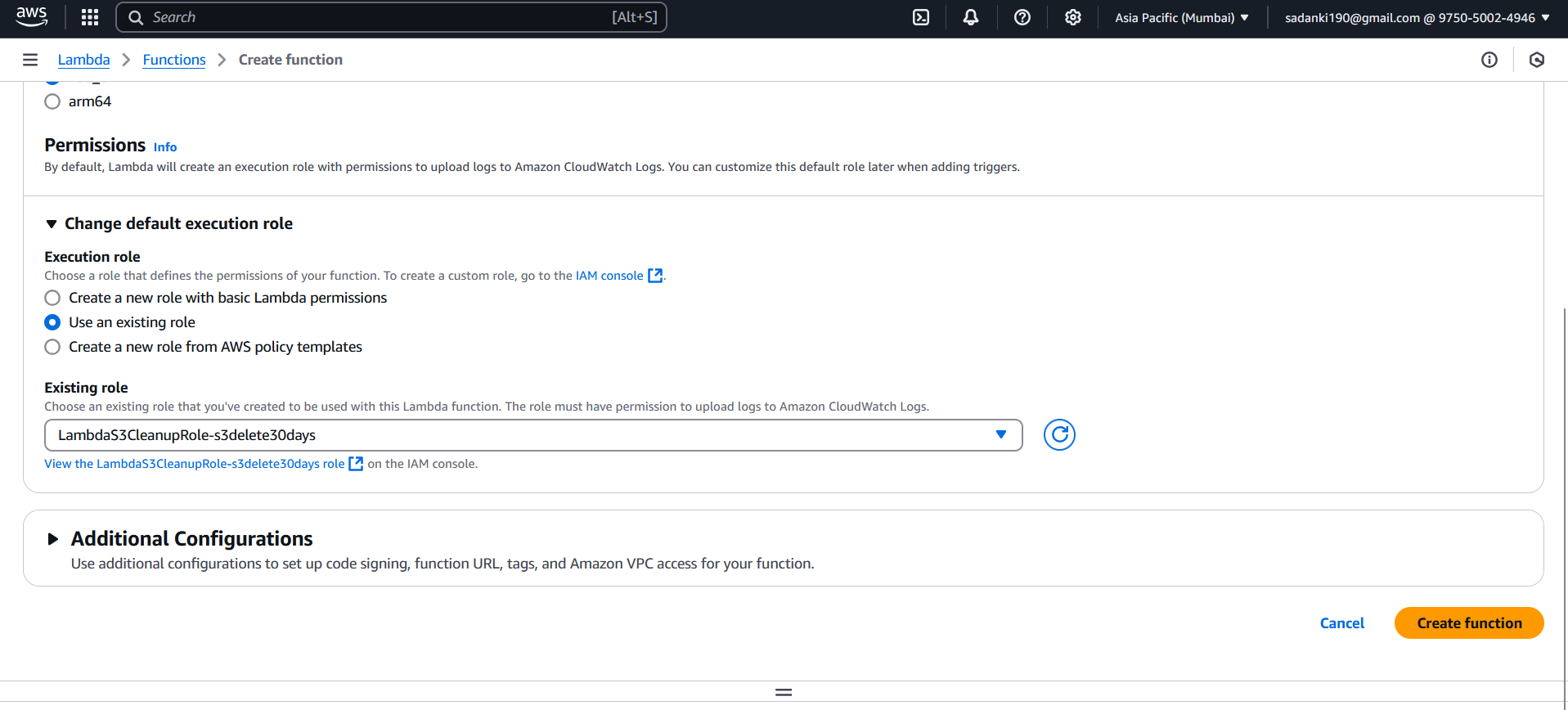


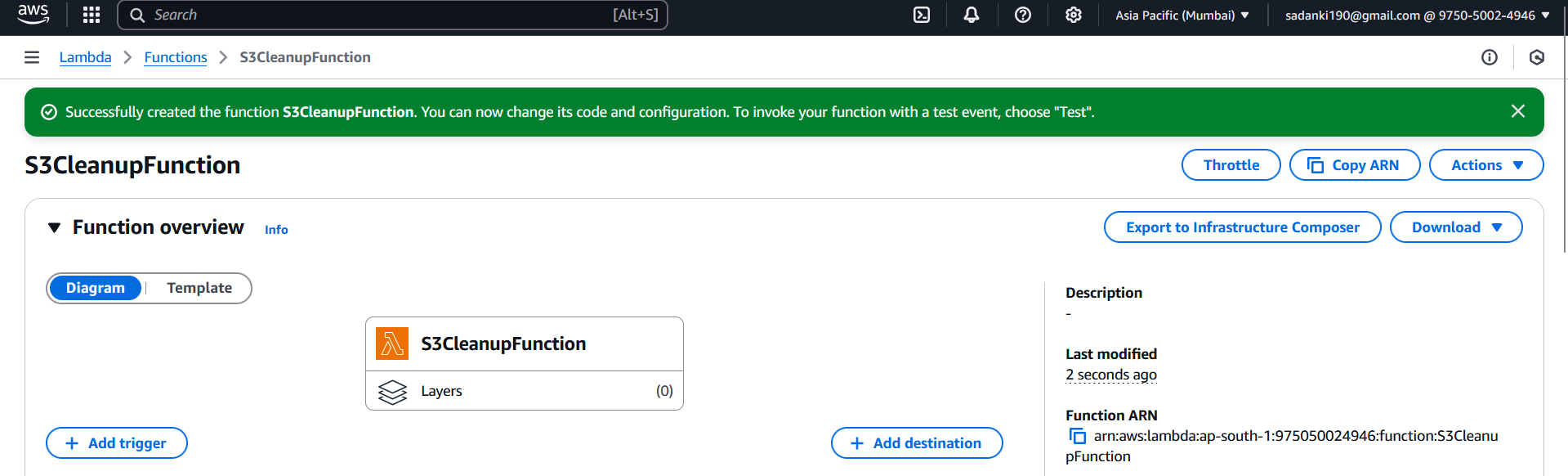


**Step 4: Create the Lambda Function**

1. Go to **AWS Lambda → Create function**.
2. Choose **Author from scratch**.
3. Enter function name: S3CleanupFunction.
4. **Runtime**: Select Python 3.12 (or latest version).
5. Under **Permissions**, select **Use an existing role**.
6. Choose the role LambdaS3CleanupRole-s3delete30days you created earlier.
7. Click **Create function**.

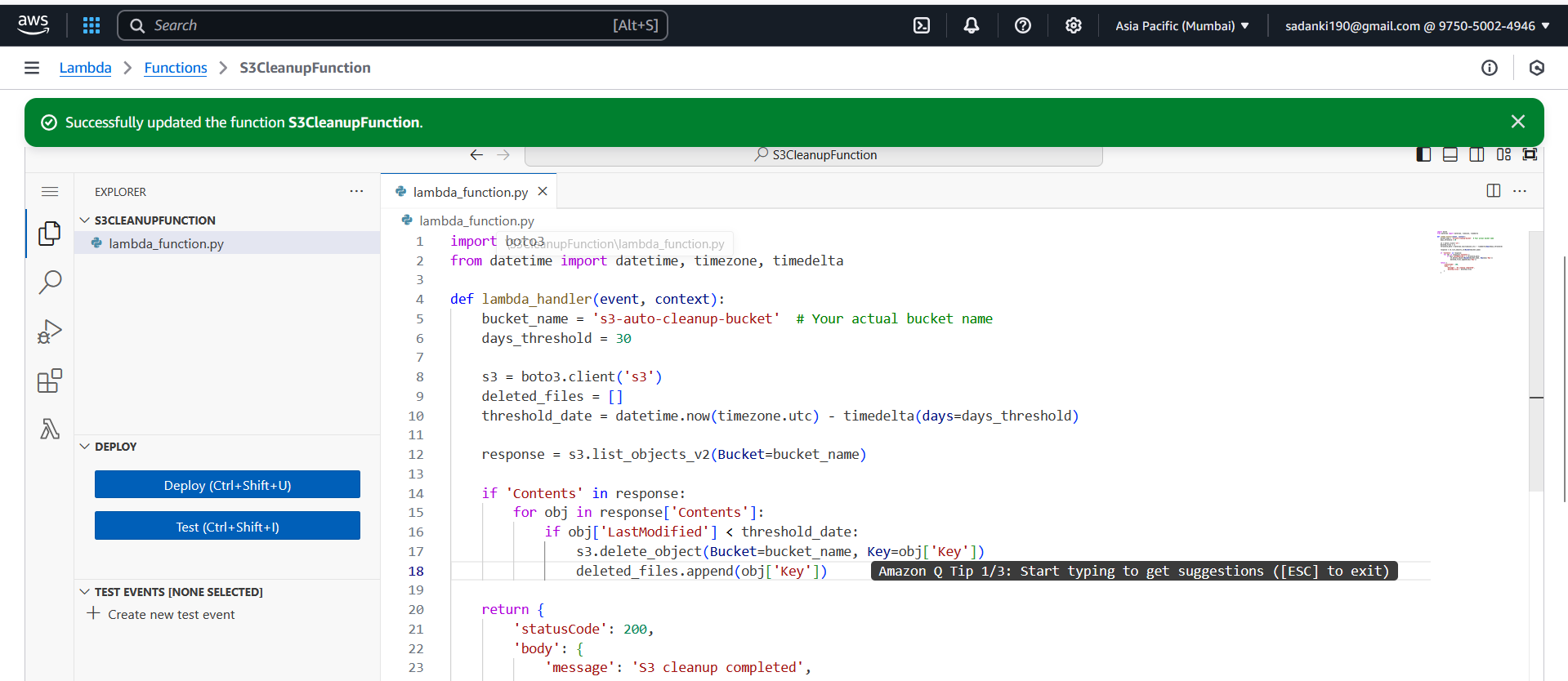


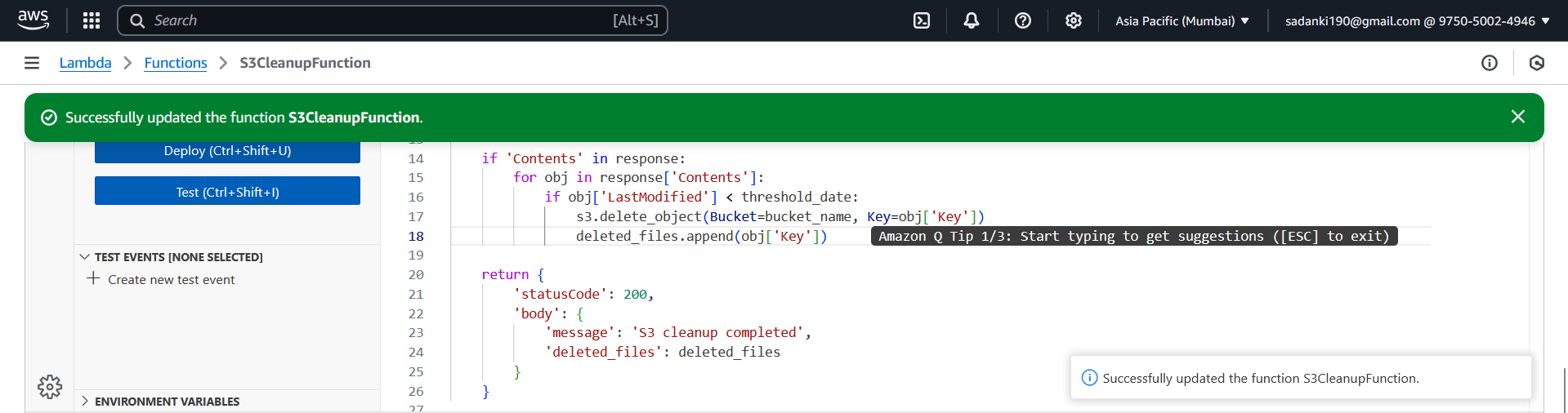




**Step 5: Add the Cleanup Code**

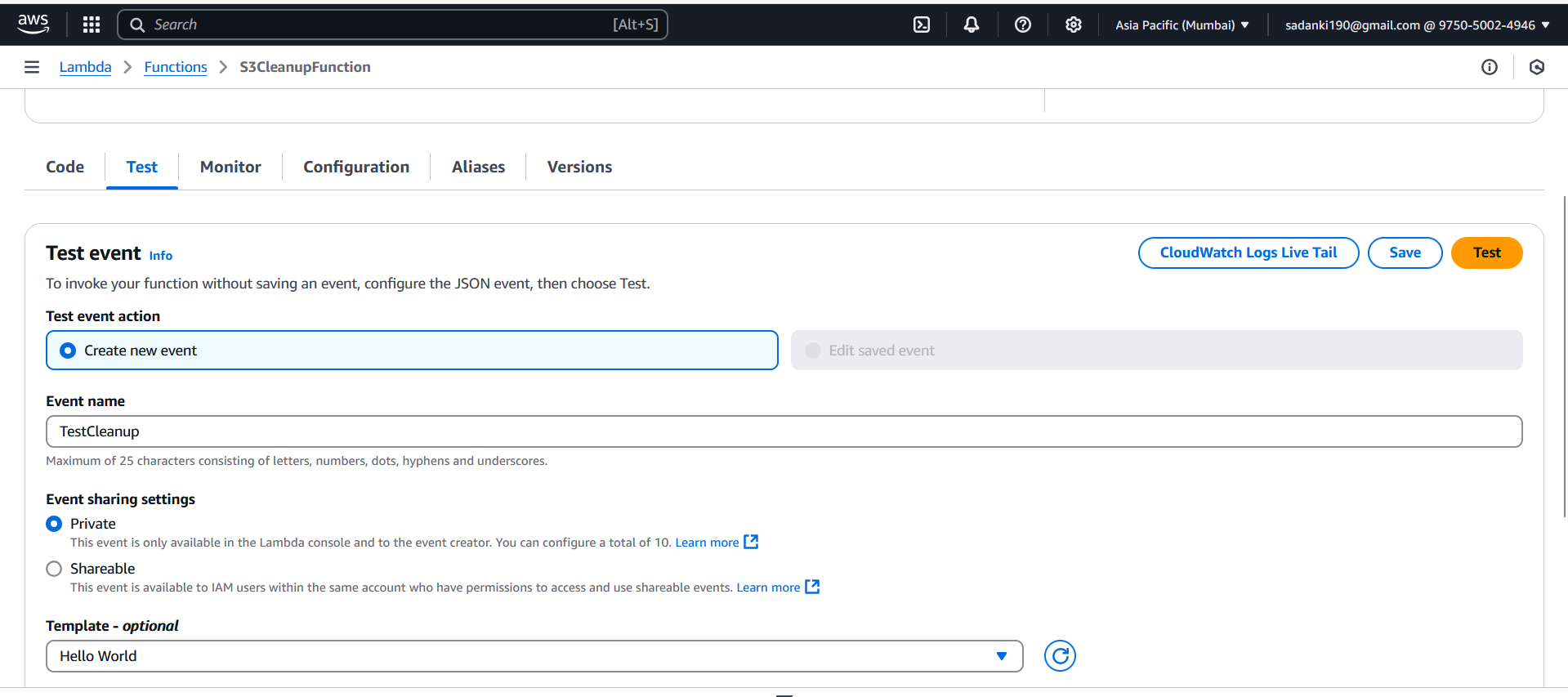
1. Scroll down to the **Code** section of your Lambda function.
2. Replace the default code with the following:
3. import boto3
4. from datetime import datetime, timezone, timedelta
5. def lambda\_handler(event, context):
6. bucket\_name = 's3-auto-cleanup-bucket'  # Your actual bucket name
7. days\_threshold = 30
8. s3 = boto3.client('s3')
9. deleted\_files = []
10. threshold\_date = datetime.now(timezone.utc) - timedelta(days=days\_threshold)
11. response = s3.list\_objects\_v2(Bucket=bucket\_name)
13. if 'Contents' in response:
14. for obj in response['Contents']:
15. if obj['LastModified'] < threshold\_date:
16. s3.delete\_object(Bucket=bucket\_name, Key=obj['Key'])
17. deleted\_files.append(obj['Key'])
18. return {
19. 'statusCode': 200,
20. 'body': {
21. 'message': 'S3 cleanup completed',
22. 'deleted\_files': deleted\_files
23. }
24. }

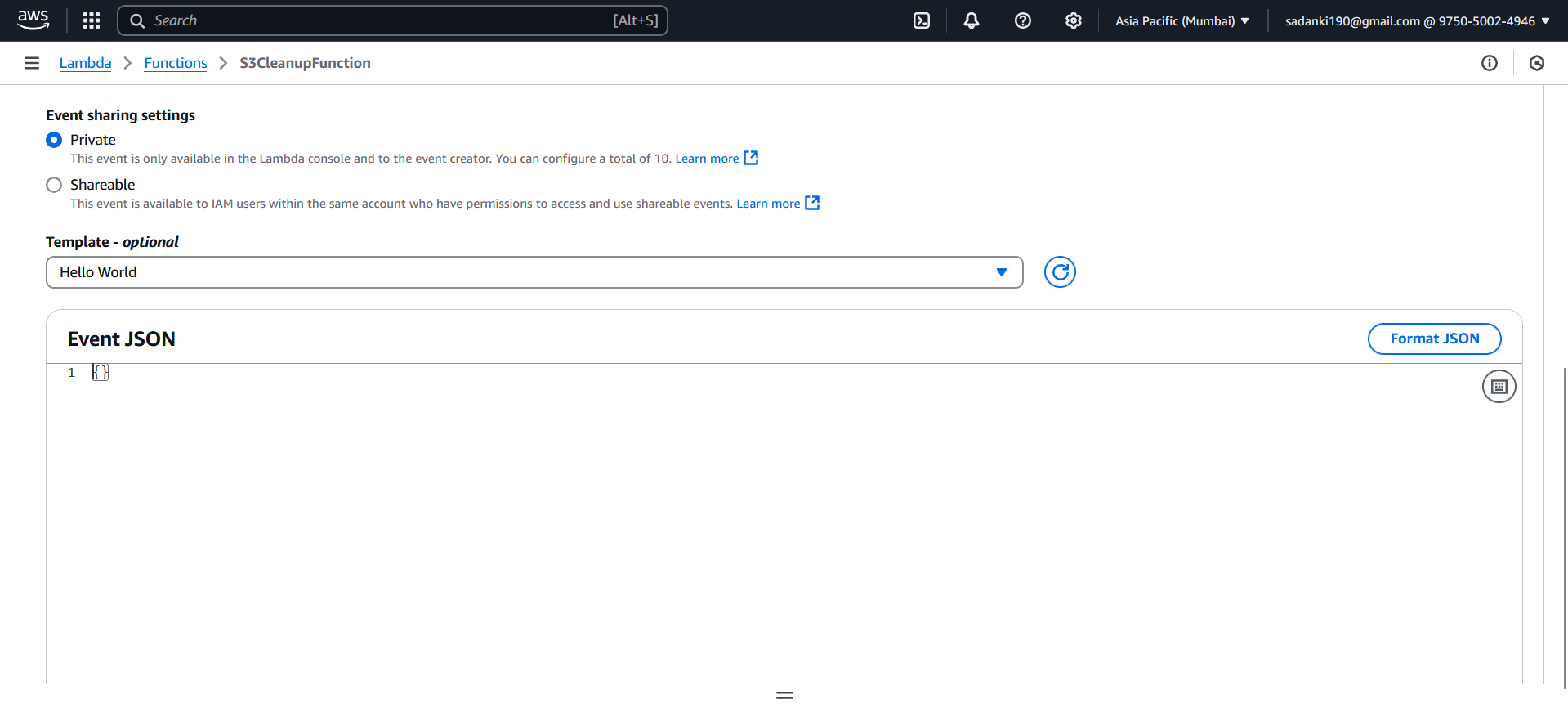




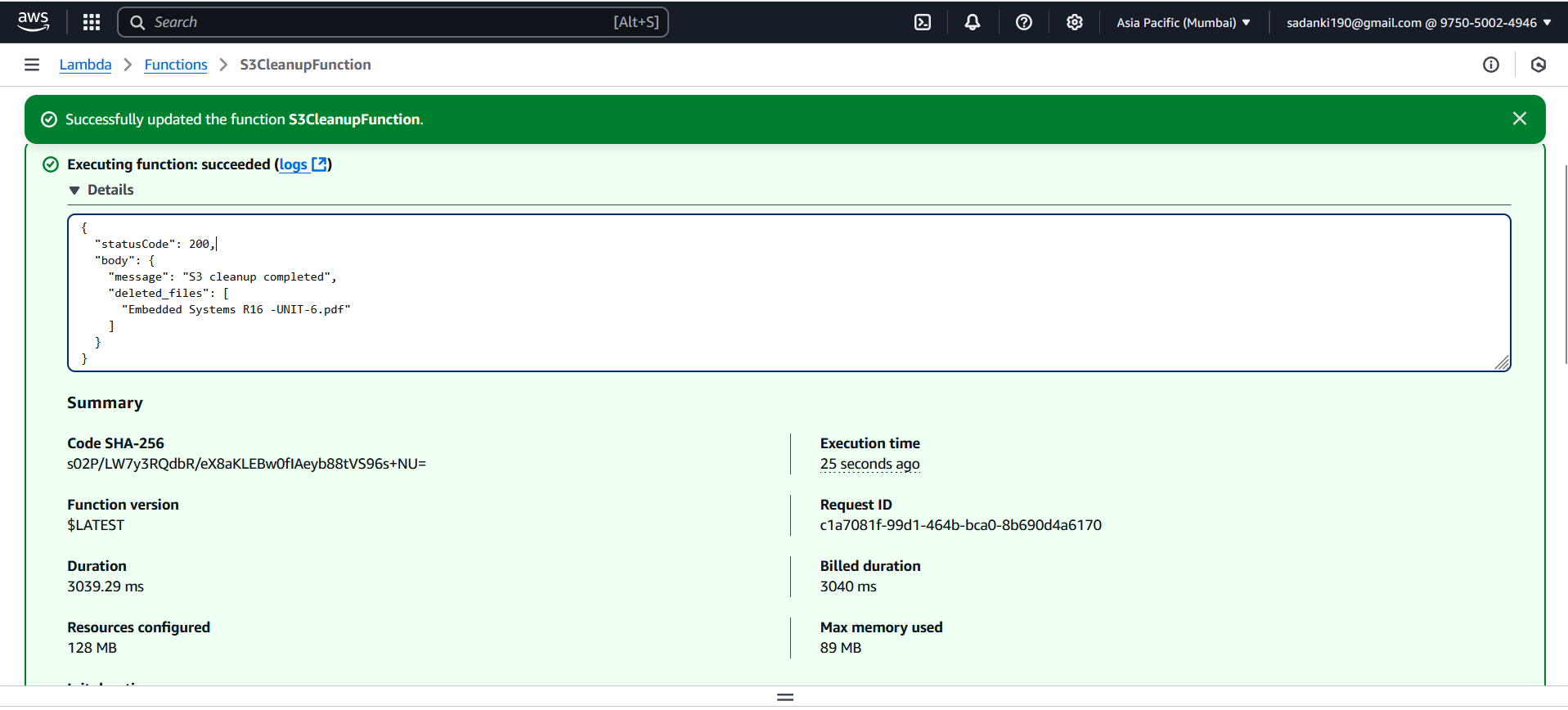
**Step 6: Test the Lambda Function**

1. Click on **Test** (top-right).
2. Create a test event:
   * Event name: TestCleanup
   * Leave the event JSON as {}.
3. Click **Test** again.





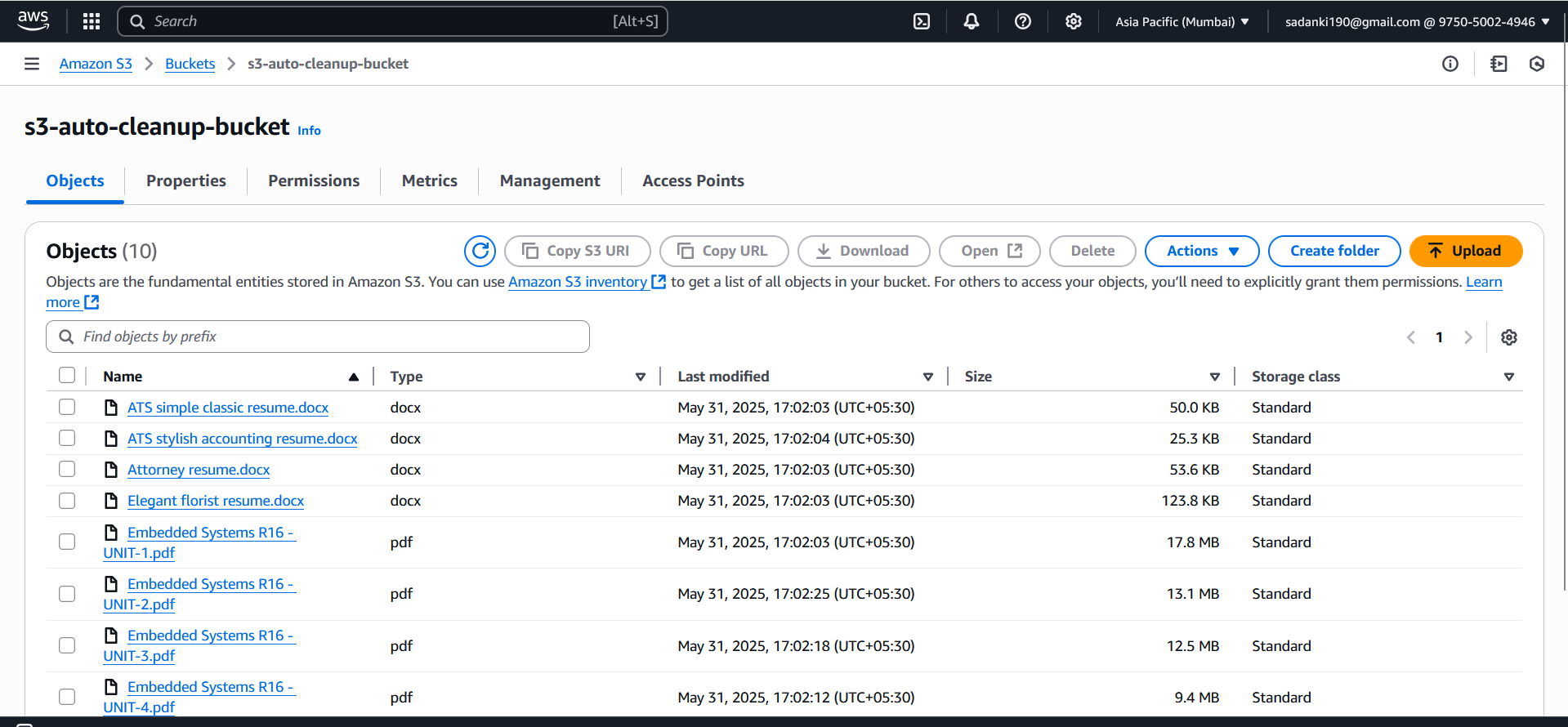
1. You’ll see the output:
   * List of deleted files (if any are older than 30 days).
   * If no files were old enough, deleted\_files will be empty.



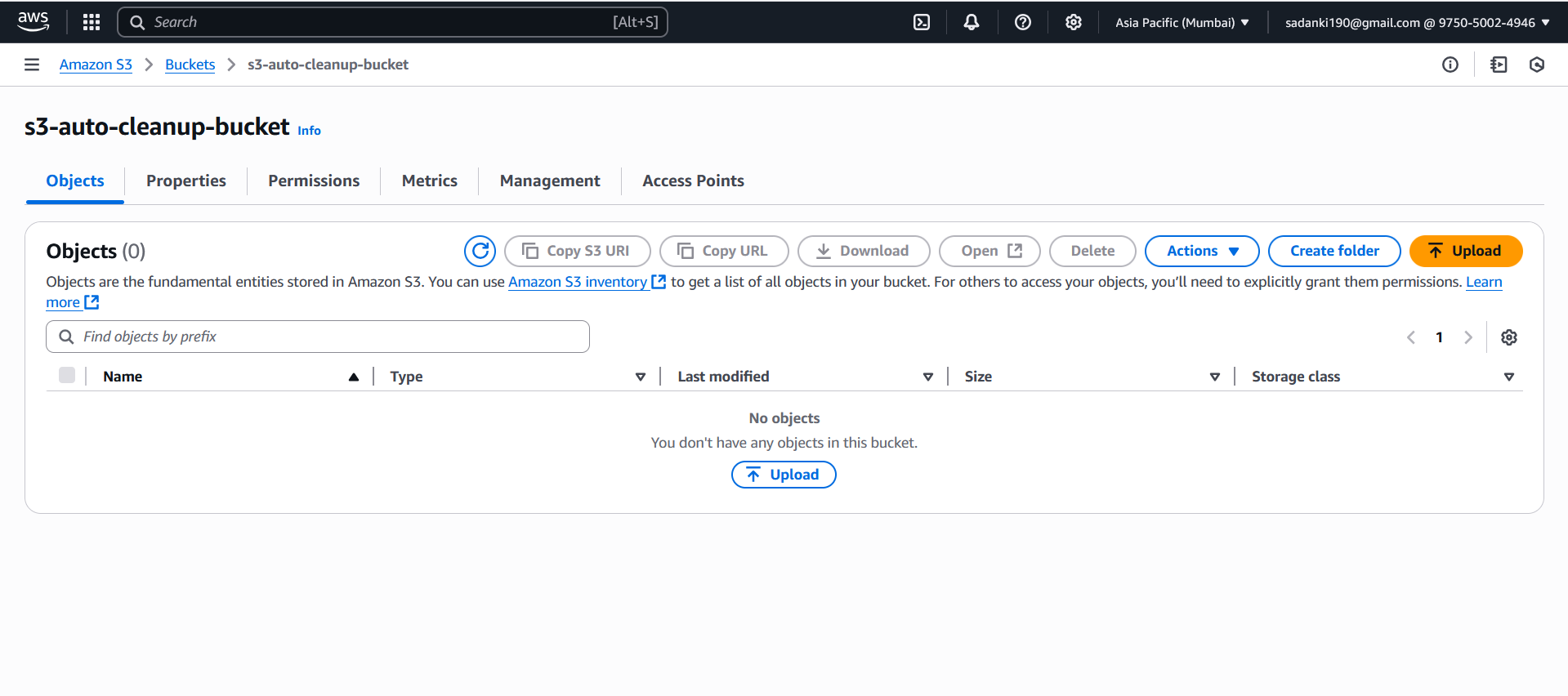
**Step 7: Verify in S3**

1. Go back to your S3 bucket.
2. Refresh the file list.
3. Files older than 30 days should be gone.
4. Files newer than 30 days should remain.

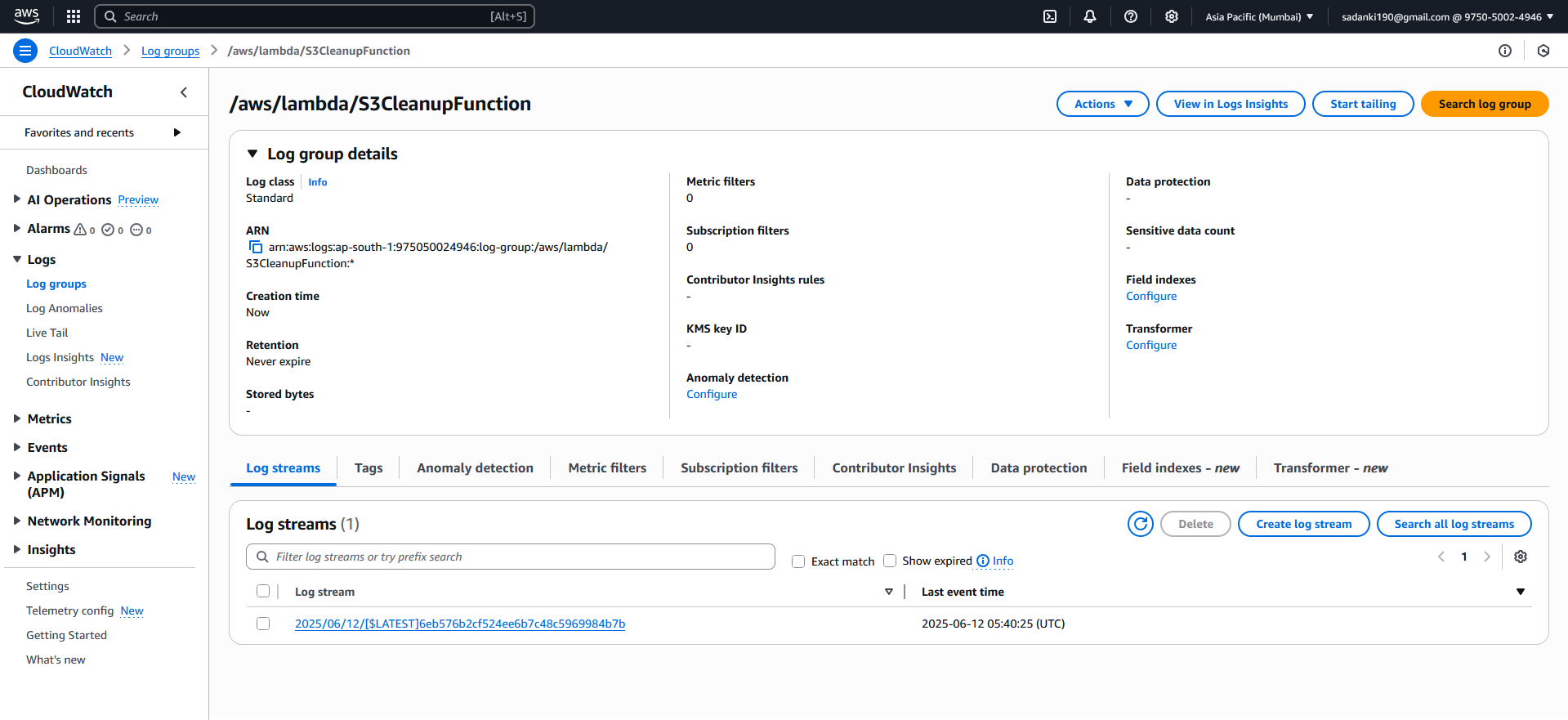
Currently we have these files in S3 bucket.

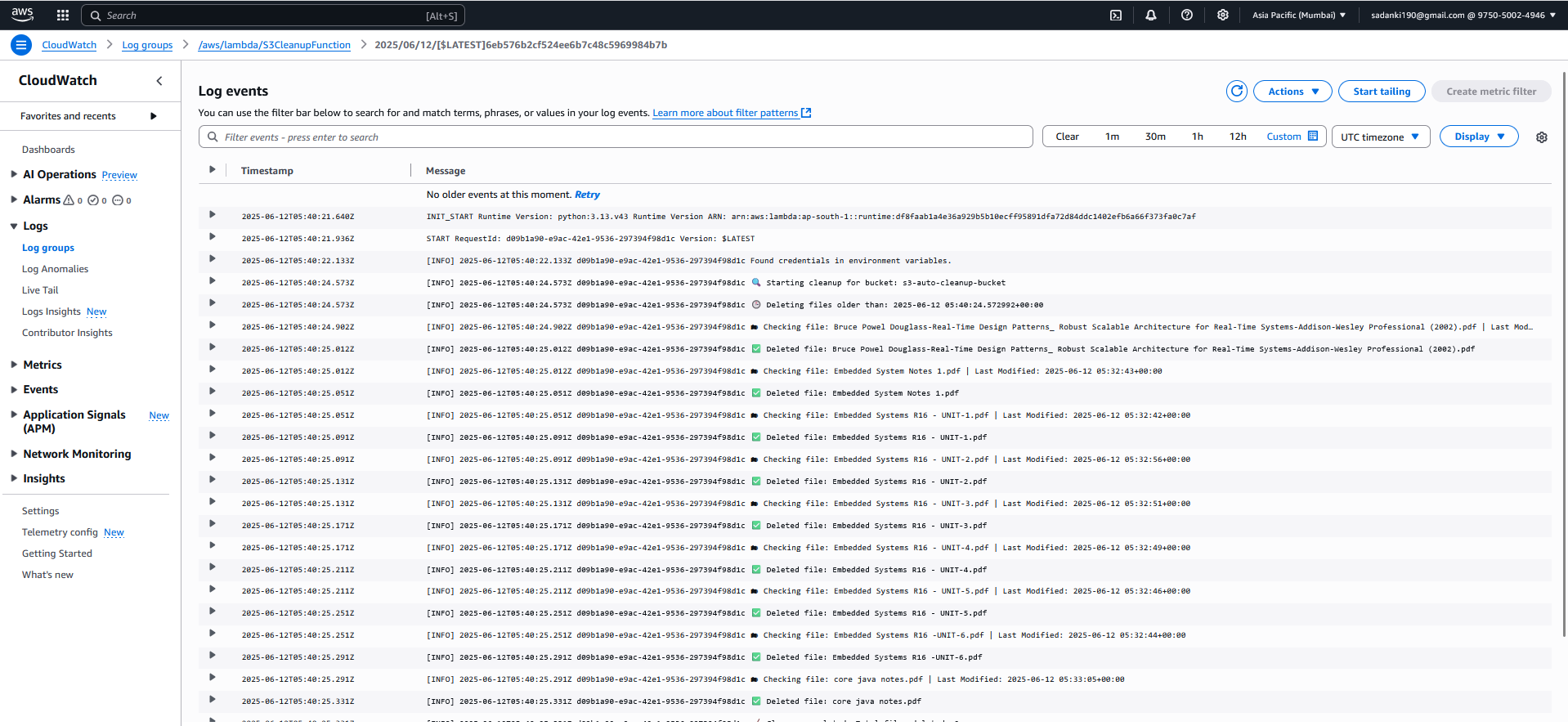


After refresh.



Logs from [**CloudWatch**](https://ap-south-1.console.aws.amazon.com/cloudwatch/home?region=ap-south-1)**:**

****

****

**Assignment 2: Automated S3 Bucket Cleanup Using AWS Lambda and Boto3 in simple terms, explain why it's useful, and cover its advantages and disadvantages.**

**Assignment Summary**

**Goal:**

**Create an AWS Lambda function using Boto3 (Python SDK for AWS) that:**

* **Connects to a specified S3 bucket**
* **Finds all files older than 30 days**
* **Deletes those old files automatically**

**Why Do This Project? (Purpose & Use Case)**

**Automated Data Lifecycle Management**

* **S3 can accumulate thousands of files over time.**
* **Some of these may be temporary, outdated, or no longer needed.**

**Use Cases:**

* **Logs, reports, backups, or temporary files that are only useful for a short time**
* **Environments where manual cleanup isn't practical**
* **Compliance with data retention policies (e.g., delete PII after 30 days)**
* **Cost reduction: You pay for storage in S3 — deleting old data saves money**

**Benefits / Advantages**

| **Advantage** | **Description** |
| --- | --- |
| **Automation** | **No manual effort — runs automatically on a schedule** |
| **Cost-effective** | **Frees up space, reduces S3 storage costs** |
| **Security & Compliance** | **Ensures data is deleted after a policy-defined time** |
| **Serverless** | **Uses AWS Lambda — no need to maintain infrastructure** |
| **Time Saving** | **Especially useful for buckets with thousands of files** |

**Disadvantages / Limitations**

| **Disadvantage** | **Description** |
| --- | --- |
| **Risk of Data Loss** | **If logic is incorrect, it might delete important files** |
| **File Metadata Limit** | **LastModified refers to S3 upload time — not the original file creation date** |
| **Testing Required** | **Needs proper testing to avoid accidental deletion** |
| **Limited Feedback** | **Without logging (e.g., CloudWatch), you won’t know what was deleted** |
| **Lambda Timeout** | **For large buckets, if too many files exist, Lambda might timeout (default: 3s to 15 mins)** |

**Example Scenarios**

**Useful**

* **A company uploads daily logs, but only needs to retain the last 30 days.**
* **A backup system that keeps only the most recent month's files.**
* **A data lake where old datasets become obsolete quickly.**

**Not Ideal**

* **A bucket where files must be reviewed manually before deletion**
* **If original creation dates (not upload dates) are critical**

**What You Learn from This Project**

| **Skill** | **Gained Experience** |
| --- | --- |
| **AWS Lambda** | **Writing serverless functions** |
| **Python Boto3** | **Using Python to automate AWS** |
| **IAM** | **Creating secure roles with least privilege** |
| **S3** | **Working with object storage programmatically** |
| **Logging** | **Understanding Lambda logs via CloudWatch** |

**Summary**

**This project simulates a real-world DevOps/CloudOps task that’s:**

* **Practical**
* **Reusable across many projects**
* **Important for automation, cost control, and data hygiene**