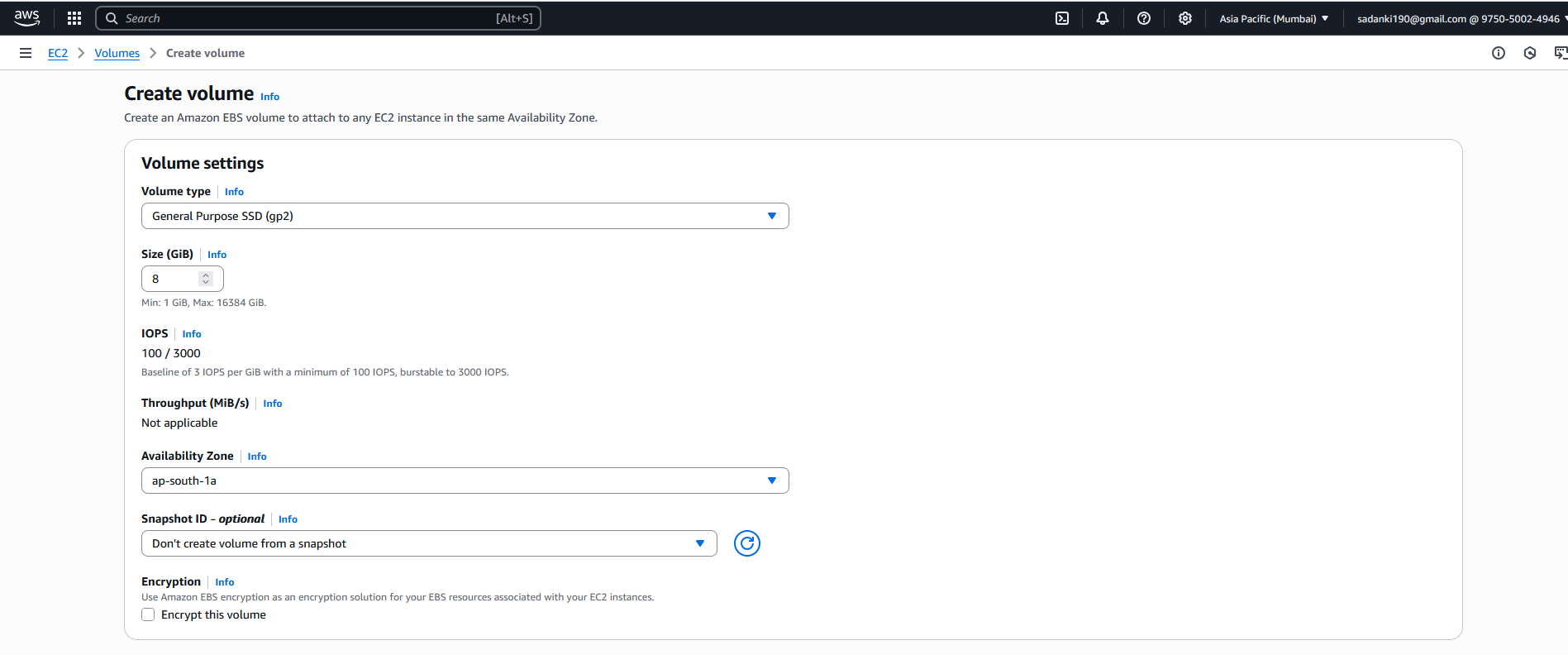
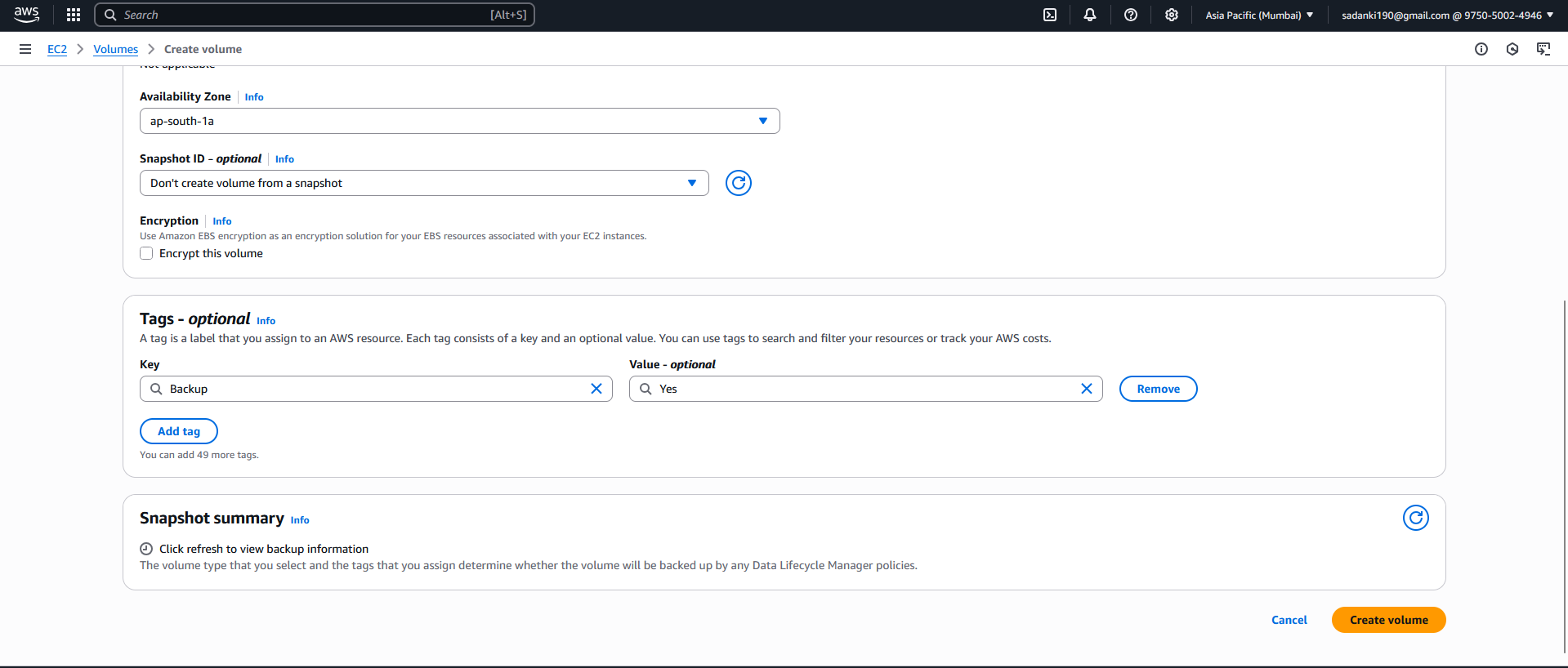
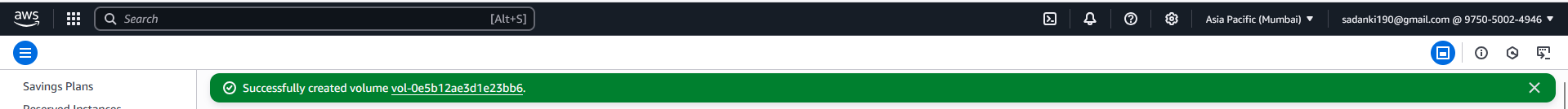
Step 1: EBS Volume Setup

1. Go to **EC2 Dashboard** → **Elastic Block Store** → **Volumes**.
2. If you don’t have one:
   * Click **Create Volume**.
   * Choose Volume Type (e.g., gp2), Size (e.g., 8 GiB), and Availability Zone.
   * Click **Create Volume**.
3. Note down the **Volume ID** (e.g., vol-0123abcd...).

| **Setting** | **Value** |
| --- | --- |
| **Volume Type** | General Purpose SSD (gp2) |
| **Size (GiB)** | 8 (well within free tier) |
| **Availability Zone** | Select any AZ (e.g., ap-south-1a if you're in Mumbai region) |
| **IOPS / Throughput** | Leave default |
| **Snapshot ID** | Leave empty |
| **Encryption** | **Uncheck "Encrypt this volume"** |
| **Tags** | (Optional) Add a tag: Key=Backup, Value=Yes |







Step 2: Create IAM Role for Lambda

**Step-by-step Instructions:**

**1. Go to the IAM Console:**

* In the AWS Console, search for **IAM** and open the IAM dashboard.
* In the left panel, click on **"Roles"**.
* Click the **“Create role”** button.

**2. Select Trusted Entity:**

* **Trusted entity type**: Select **AWS service**
* **Use case**: Choose **Lambda**
* Click **Next**.

**Attach Permissions Policies:**

To keep it simple for your college project, you can attach a pre-built policy:

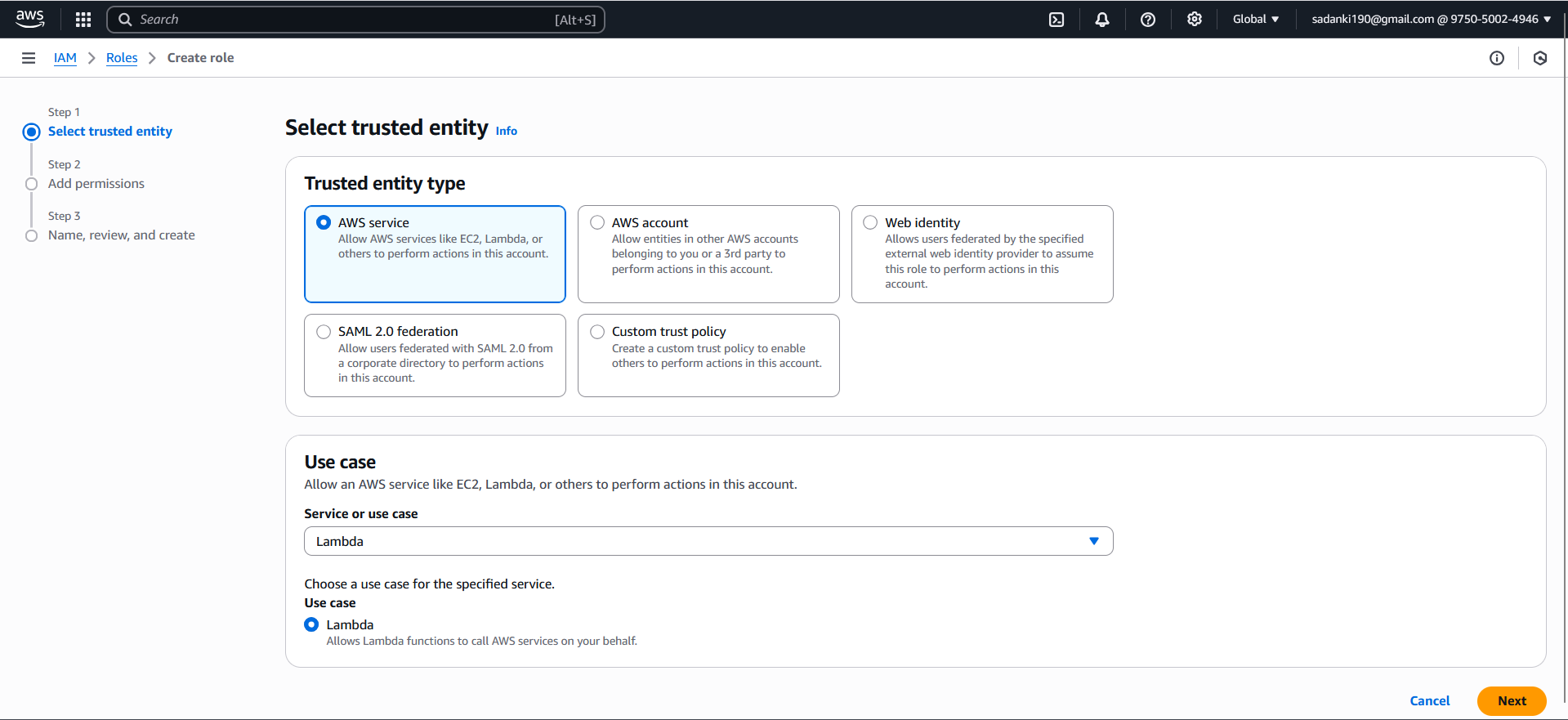
* In the **search bar**, type:
  + ✅ AmazonEC2FullAccess – This allows snapshot creation and deletion.
  + Select the checkbox next to AmazonEC2FullAccess
  + Click **Next**.

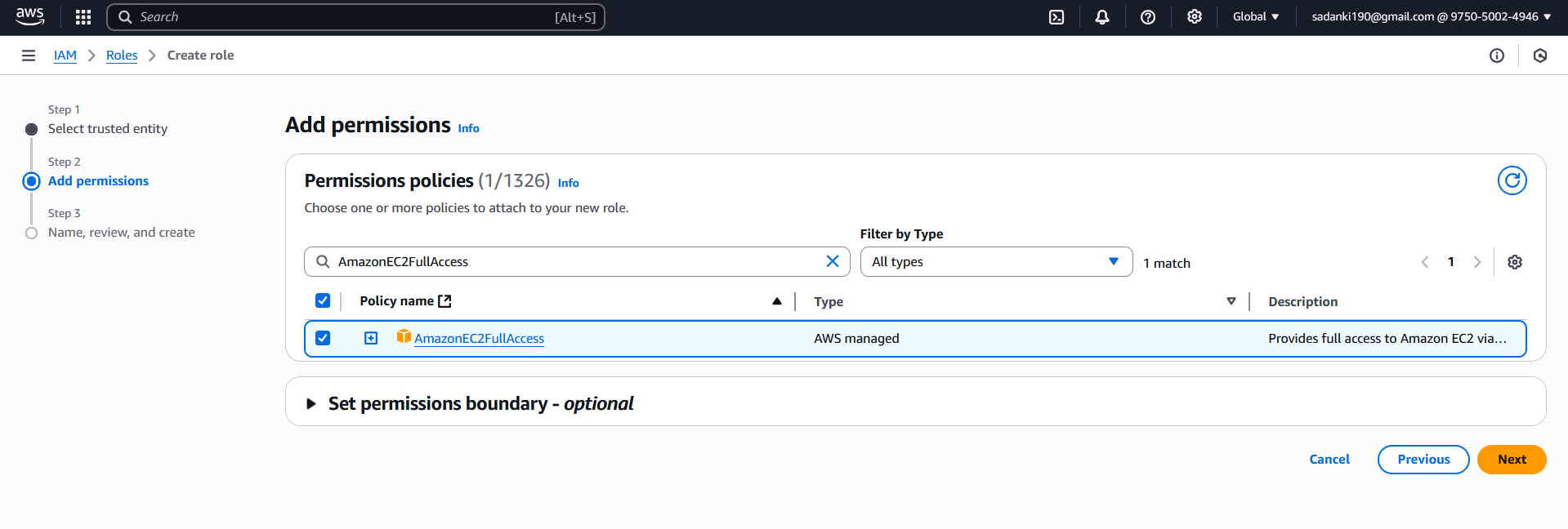
**4. Name and Tag the Role:**

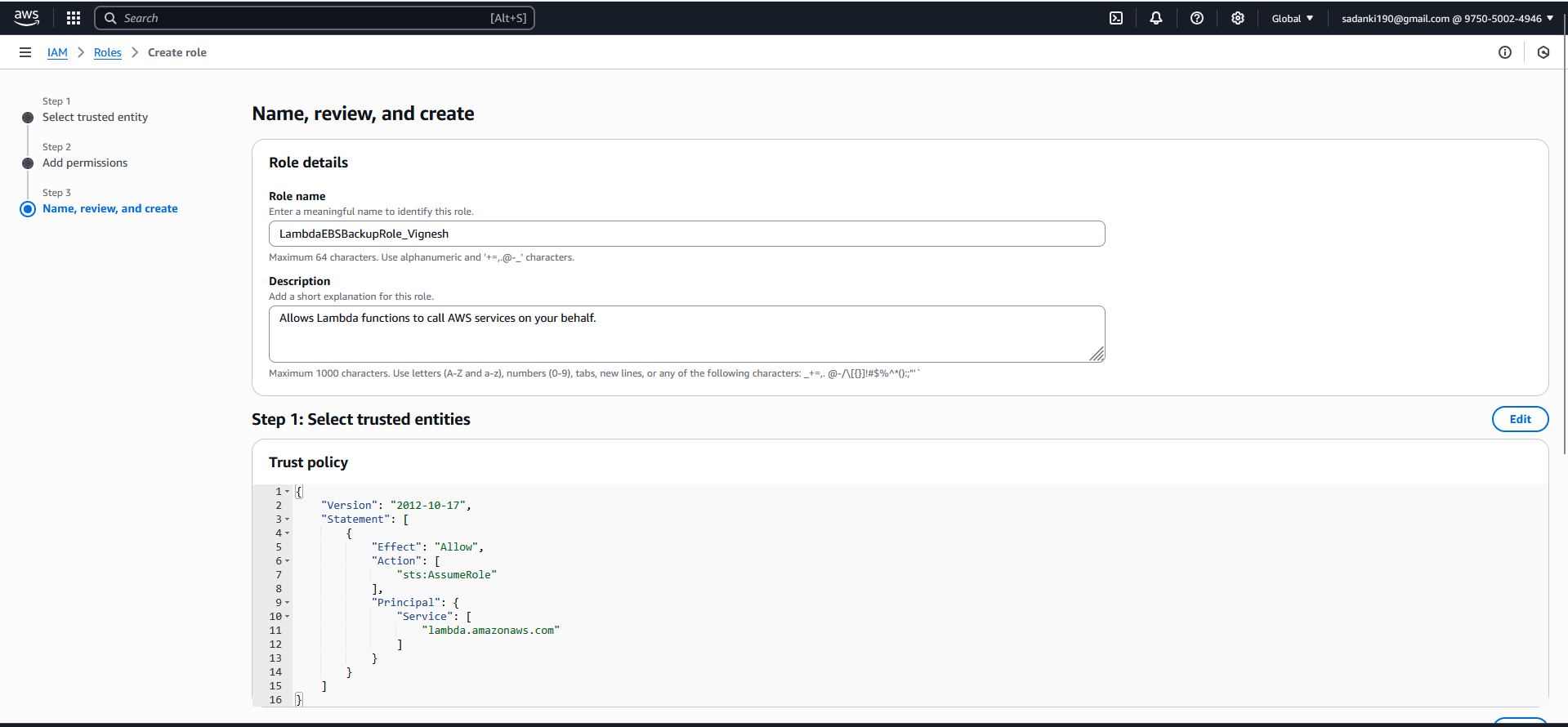
* **Role Name**: LambdaEBSBackupRole
* (Optional) Add tags like:
  + Key: Project, Value: EBSBackupAutomation
* Click **Create Role**

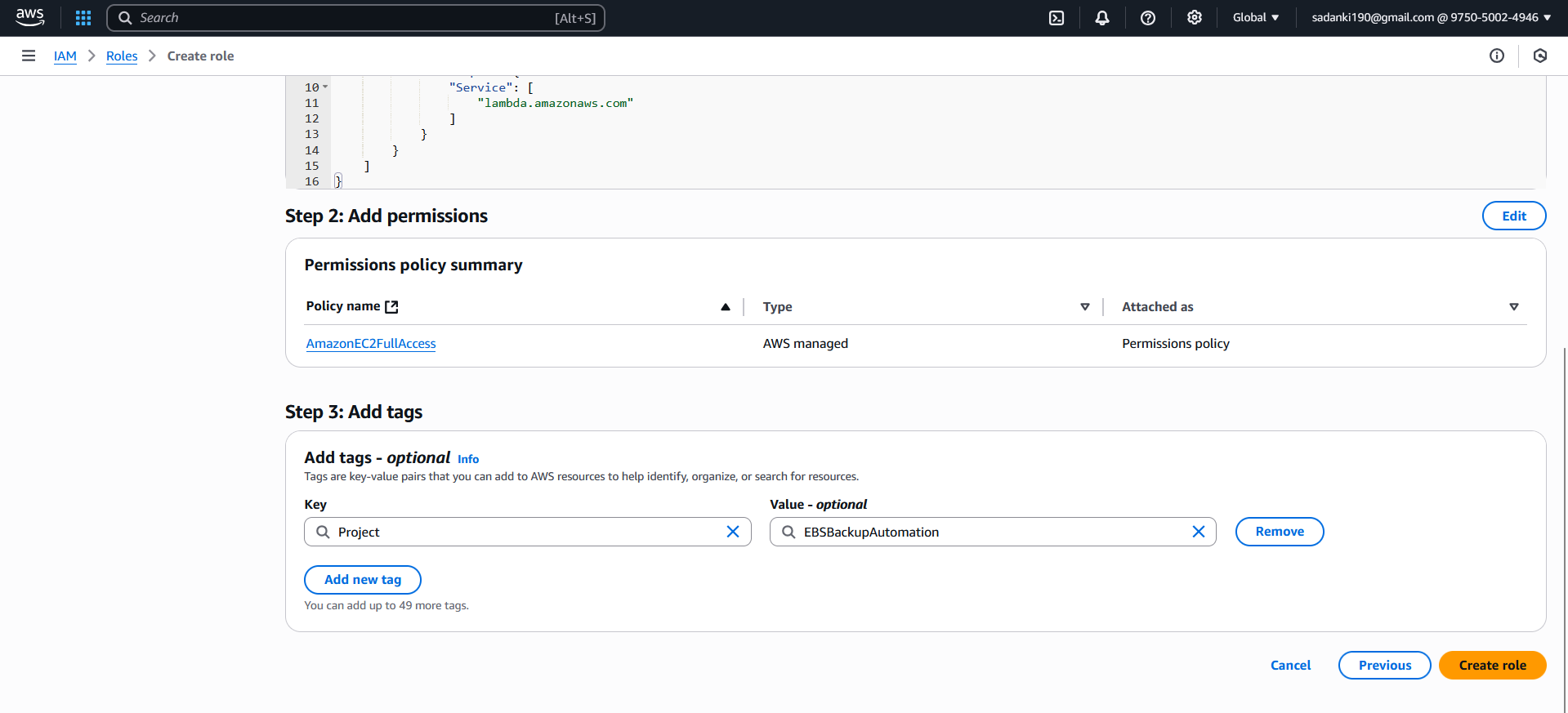
**Confirmation:**

* You’ll see your new role LambdaEBSBackupRole in the list.
* Click it and **copy the Role ARN** — you’ll need this when assigning the role to your Lambda function.











Step 3: Create the Lambda Function (Python, Boto3)

**1. Go to Lambda Console:**

* In the AWS Console, search for **Lambda** and open it.
* Click **“Create function”**.

**2. Basic Lambda Setup:**

* **Function name**: EBSBackupAutomation
* **Runtime**: Python 3.12 (or Python 3.10+)
* **Execution Role**:
  + Select **“Use an existing role”**
  + Choose the role you created earlier: LambdaEBSBackupRole
* Click **Create Function**

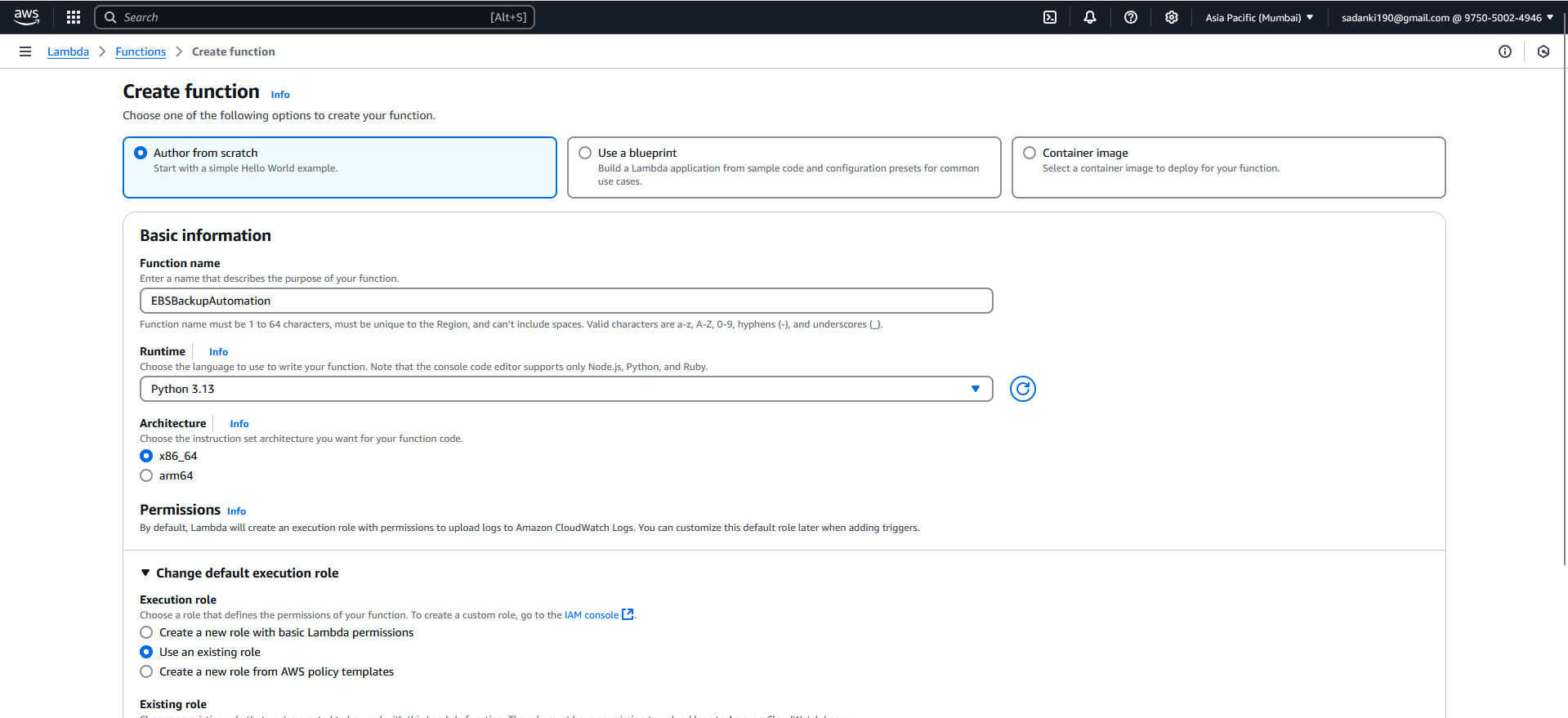
1. **Paste the Python Code**:
2. import boto3
3. import datetime
4. # Replace with your EBS Volume ID
5. VOLUME\_ID = 'vol-0e5b12ae3d1e23bb6'
6. REGION = 'ap-south-1'
7. RETENTION\_DAYS = 30
8. def lambda\_handler(event, context):
9. ec2 = boto3.client('ec2', region\_name=REGION)
10. # 1. Create snapshot
11. snapshot = ec2.create\_snapshot(
12. VolumeId=VOLUME\_ID,
13. Description=f"Automated snapshot for {VOLUME\_ID}",
14. TagSpecifications=[
15. {
16. 'ResourceType': 'snapshot',
17. 'Tags': [{'Key': 'CreatedBy', 'Value': 'LambdaBackup'}]
18. }
19. ]
20. )
21. print(f"Created snapshot: {snapshot['SnapshotId']}")
22. # 2. Get all snapshots created by this function
23. snapshots = ec2.describe\_snapshots(
24. Filters=[
25. {'Name': 'volume-id', 'Values': [VOLUME\_ID]},
26. {'Name': 'tag:CreatedBy', 'Values': ['LambdaBackup']}
27. ],
28. OwnerIds=['self']
29. )['Snapshots']
30. # 3. Delete old snapshots
31. for snap in snapshots:
32. start\_time = snap['StartTime'].replace(tzinfo=None)
33. age = (datetime.datetime.now() - start\_time).days
34. if age > RETENTION\_DAYS:
35. ec2.delete\_snapshot(SnapshotId=snap['SnapshotId'])
36. print(f"Deleted snapshot: {snap['SnapshotId']} (Age: {age} days)")

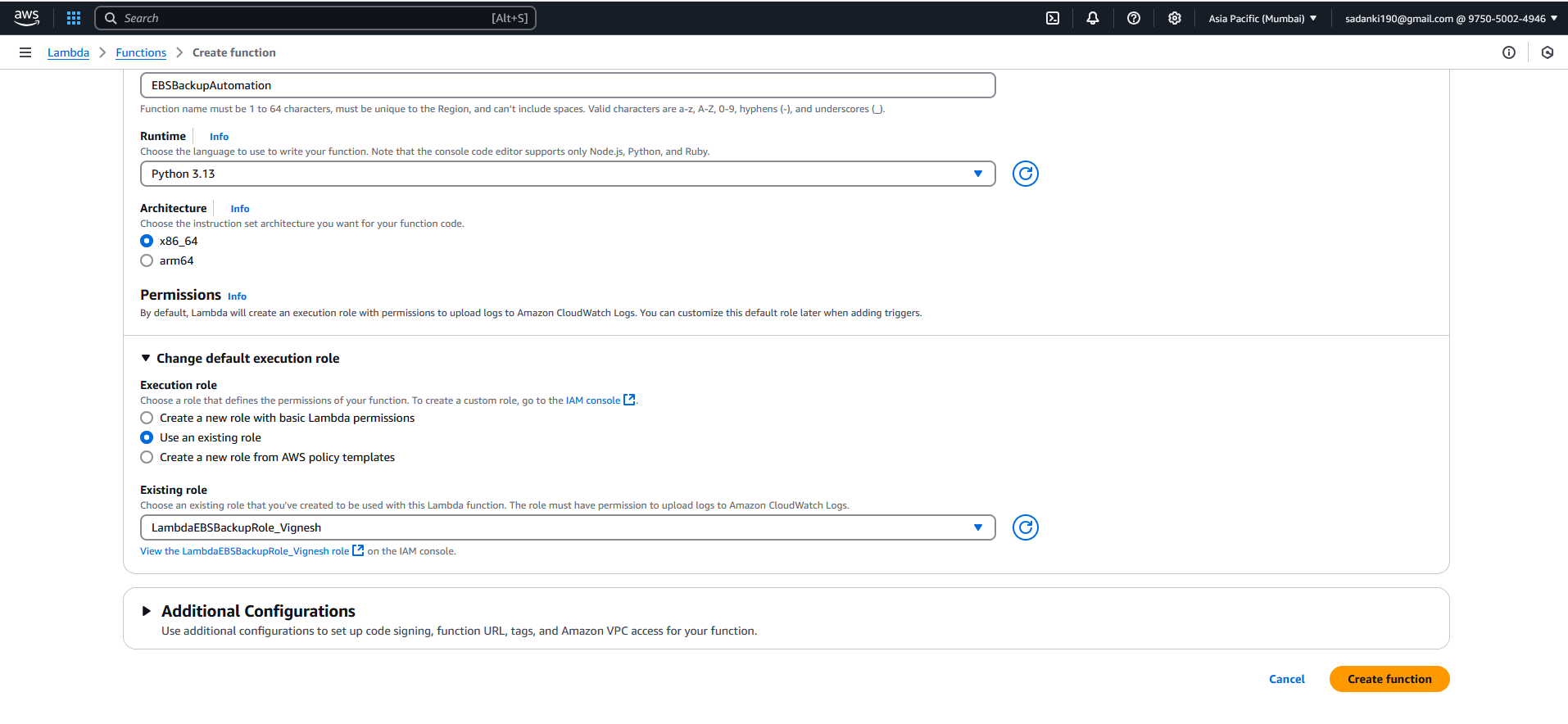
**4. Deploy the Lambda Function:**

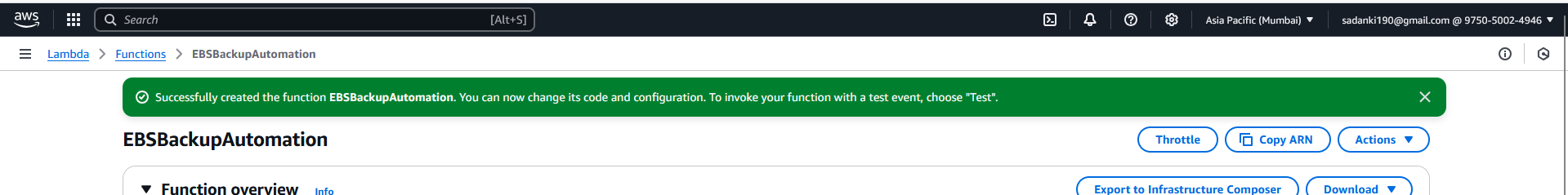
* Click **Deploy** (top-right button above code editor)

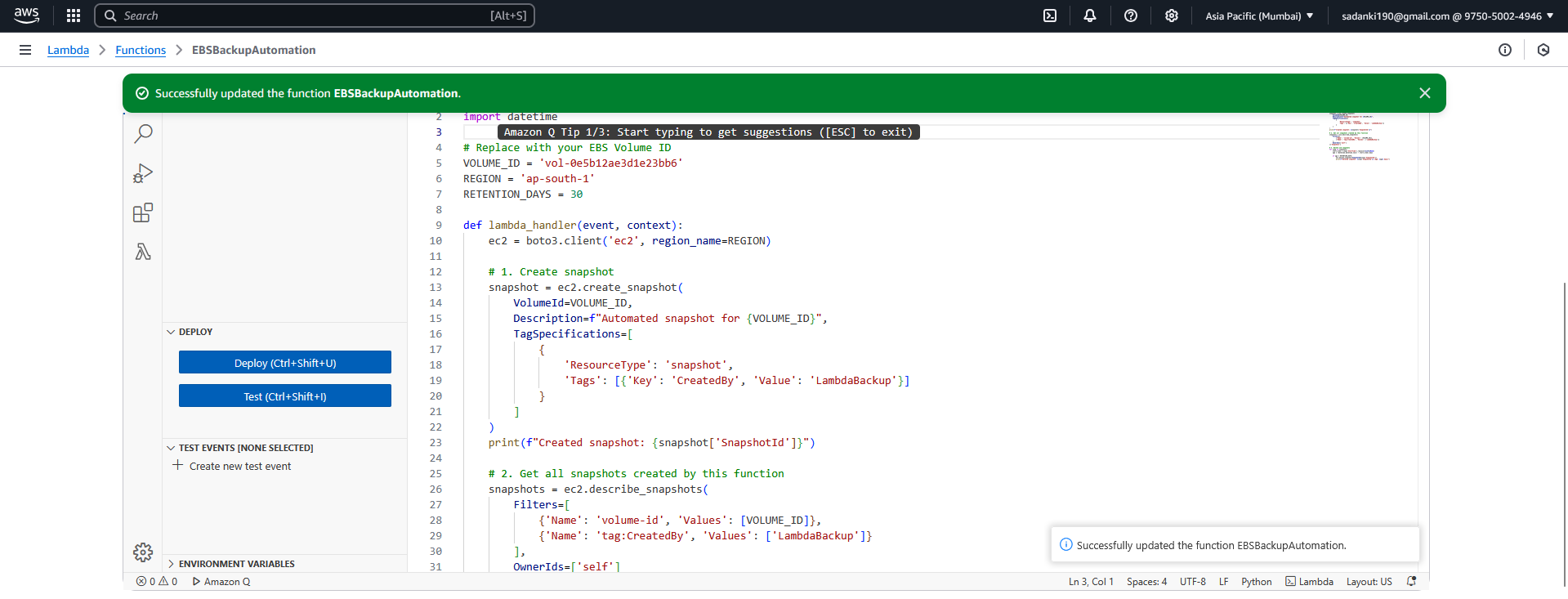
**5. Test Manually (Optional):**

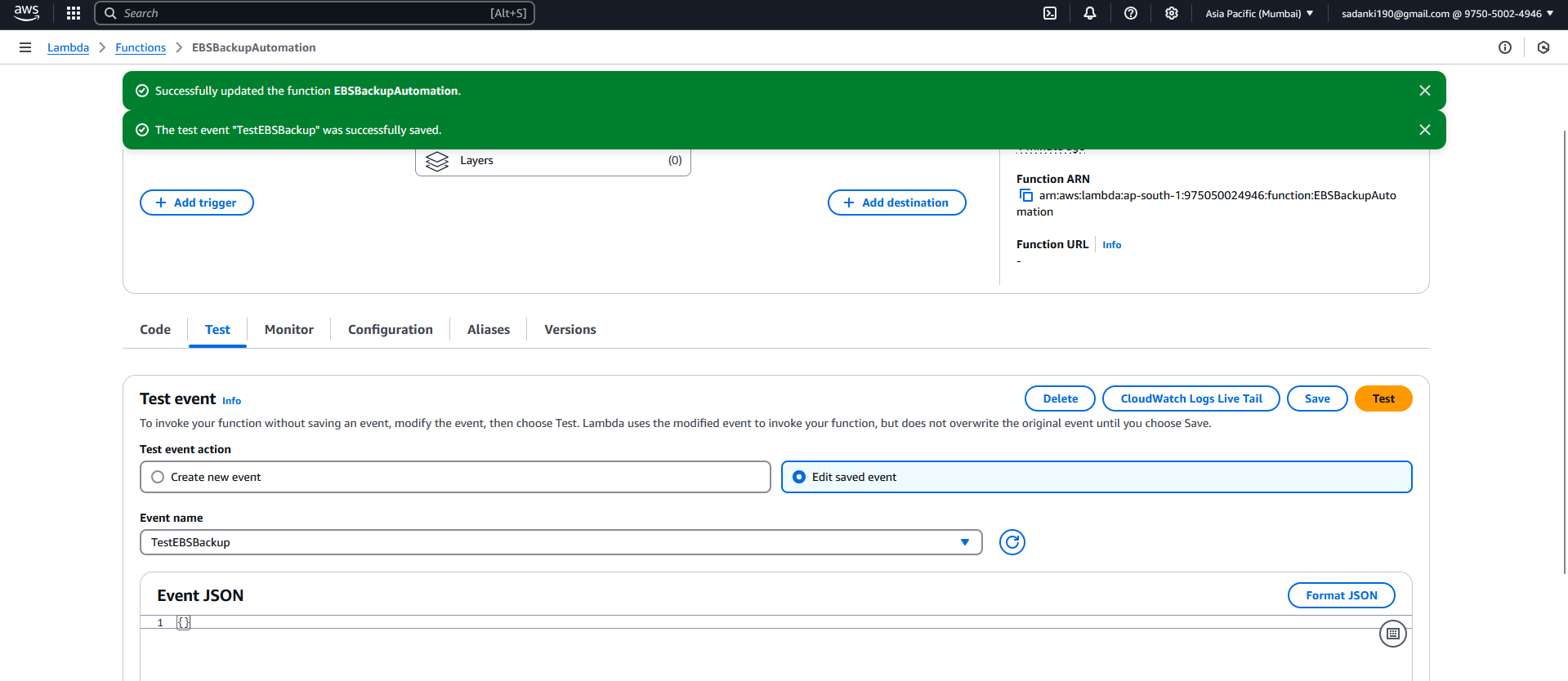
* Click **Test**
* Create a new test event → Name it TestEBSBackup
* Use the default JSON ({}) and click **Test**
* Check **Logs** in **Monitor > View Logs in CloudWatch** to verify snapshot creation

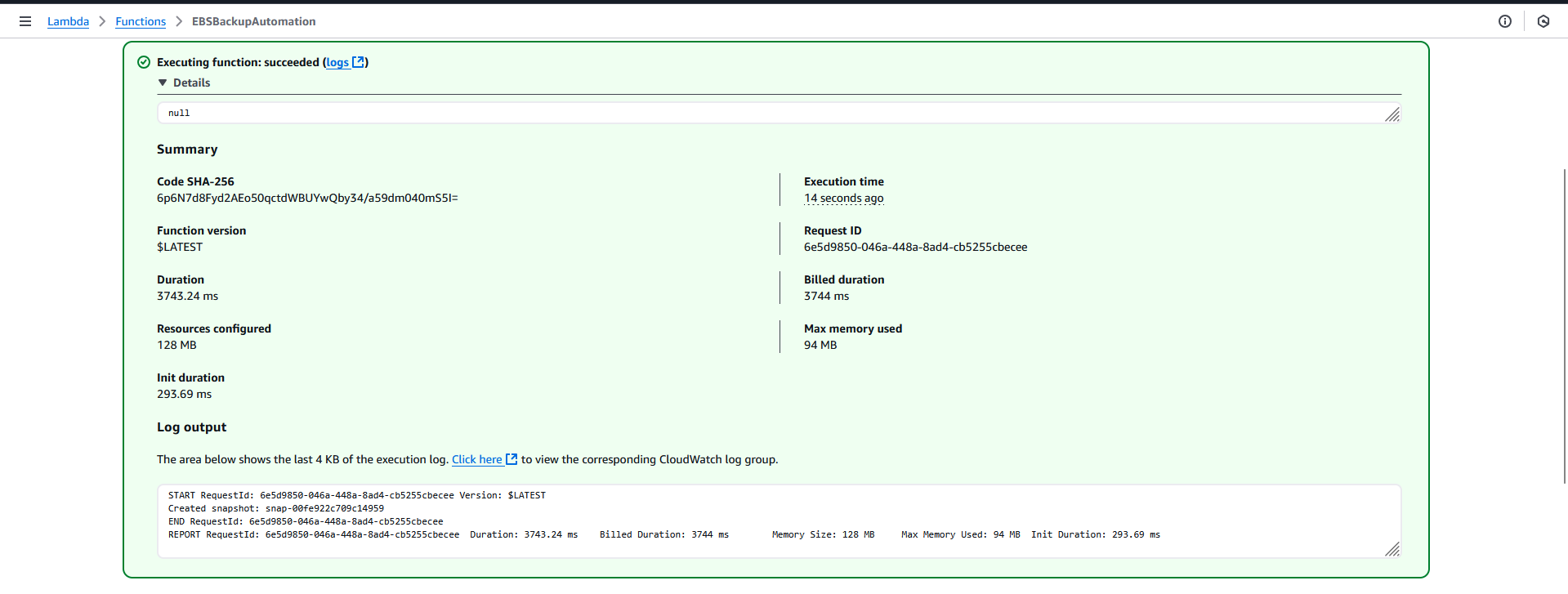


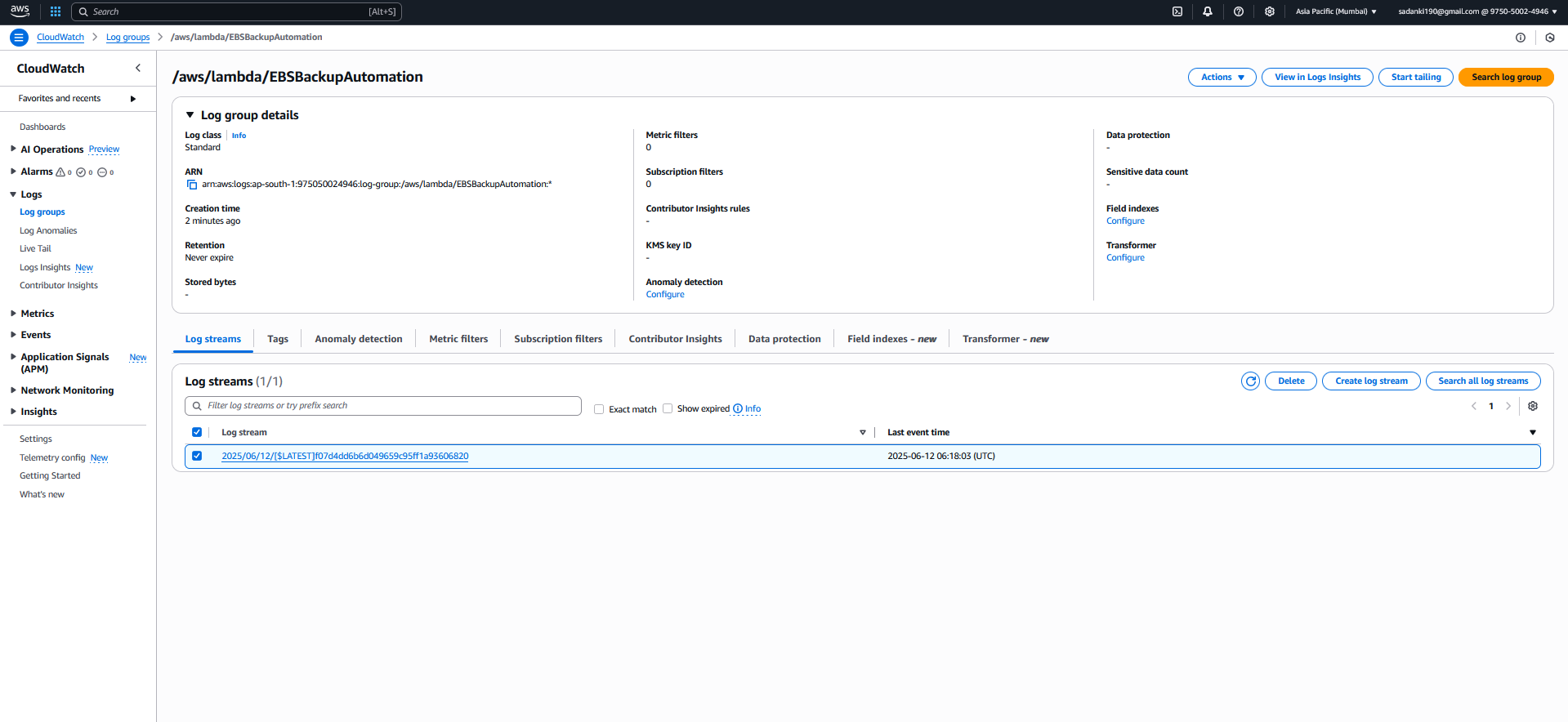


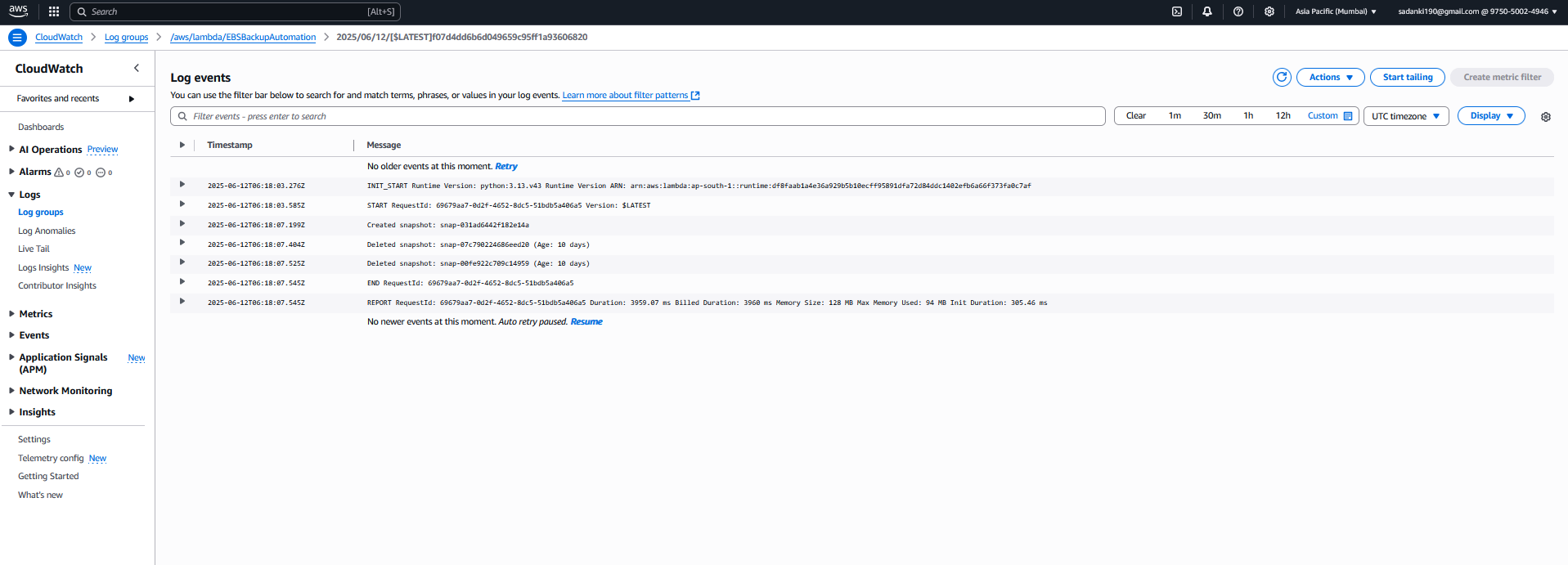








Logs from cloudwatch: 



Step 4: Set Up a CloudWatch Event Rule (Schedule Trigger)

**Steps to Create the CloudWatch Event Rule:**

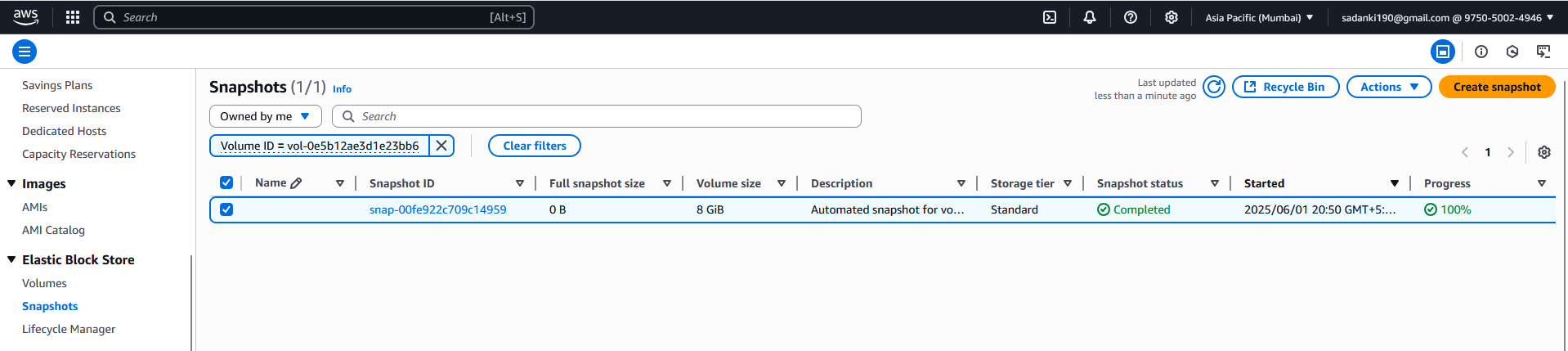
1. **Go to the CloudWatch Console**
   * Search for **"CloudWatch"** in the AWS Console.
   * Select **"Rules"** under **"Events"** or go to:  
     https://console.aws.amazon.com/cloudwatch/home#rules:
2. **Create a New Rule**
   * Click on **"Create rule"**.
3. **Step 1: Define Rule Details**
   * **Rule type**: Select **"Schedule"**
   * **Schedule pattern**: Choose **"Recurring schedule"**
   * **Period**: Select **"Week"**
     + Day: **Sunday**
     + Time: **12:00 PM**
     + Time zone: **Asia/Kolkata (IST)** or your local time
4. **Step 2: Select Target**
   * **Target type**: **AWS Service**
   * **Service**: Select **Lambda function**
   * **Function**: Choose your function → EBSBackupAutomation
5. **Step 3: Configure Tags (Optional)**
   * You can skip or add something like:
     + Key: Project
     + Value: EBSBackup
6. **Step 4: Create Rule**
   * Click **"Create rule"**

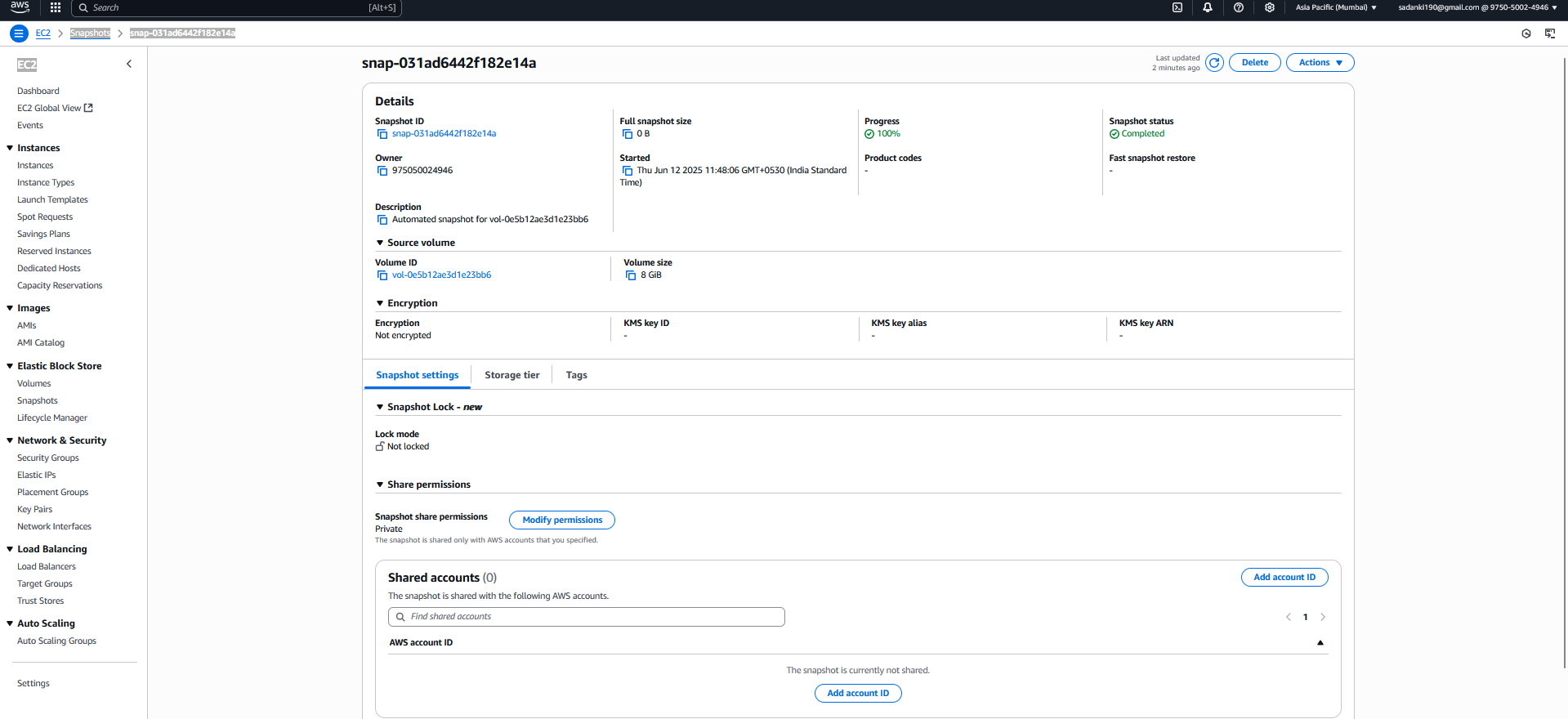
**Step 6: Final Testing and Confirmation**

Now that the Lambda function works, let’s validate everything one last time.

**🔍 Verify Snapshots Were Created:**

1. Go to **EC2 Dashboard** → **Snapshots** (left sidebar under “Elastic Block Store”).
2. Filter by:
   * Volume ID: vol-0e5b12ae3d1e23bb6
   * Tag: CreatedBy = LambdaBackup





**Assignment 4**

**Assignment 4: Automatic EBS Snapshot and Cleanup Using AWS Lambda and Boto3**

**Objective**

To automate the process of:

* Creating snapshots of an Amazon EBS volume on a schedule
* Deleting old snapshots (older than a defined retention period, e.g., 7 or 10 days)

This ensures data backup without manual effort while controlling storage costs by cleaning up outdated snapshots.

**🛠️ What Was Implemented**

1. **EBS Volume Setup**
   * Used existing EBS Volume: vol-0e5b12ae3d1e23bb6
   * Size: 8 GiB
2. **IAM Role Setup**
   * Role: LambdaEBSBackupRole
   * Policies Attached:
     + AmazonEC2FullAccess
     + AWSLambdaBasicExecutionRole (for CloudWatch logs)
3. **Lambda Function**
   * Function name: EBSBackupAutomation
   * Runtime: Python 3.x
   * Key Logic:
     + Use Boto3 to create a snapshot of a specified volume
     + List existing snapshots with a specific description or tag
     + Delete snapshots older than X days (e.g., 10 days)
   * Example log output:
   * Created snapshot: snap-031ad6442f182e14a
   * Deleted snapshot: snap-087cf92204886eed08 (Age: 10 days)
   * Deleted snapshot: snap-88ef9227d9ec14959 (Age: 10 days)
4. **CloudWatch Integration**
   * Created a **scheduled rule** to automatically trigger Lambda:
     + Runs every **Sunday at 12:00 PM IST**
     + Uses a **CloudWatch Event Rule** as the trigger

**✅ Why This Is Useful**

| **Benefit** | **Explanation** |
| --- | --- |
| 🔄 Automates EBS backups | Removes the need for manual snapshot creation |
| 🧹 Cleans up old snapshots | Prevents excessive costs from unused backups |
| 💡 Follows AWS best practices | Ensures routine backups and cost optimization |
| ☁️ Serverless execution | No EC2 required — Lambda handles it all |

**⚠️ Limitations**

| **Limitation** | **Notes** |
| --- | --- |
| 📅 Time-based deletion only | Based on snapshot creation date, not data relevance |
| ❌ No encryption by default | Snapshots are not encrypted unless specifically set |
| 💰 Charges apply if usage exceeds free tier | Especially for long-term or large volume snapshots |

**Screenshots (for submission)**

Included:

* Lambda execution result
* Snapshot list in EC2 console
* CloudWatch log output (if available)
* Scheduled rule configuration (CloudWatch EventBridge)