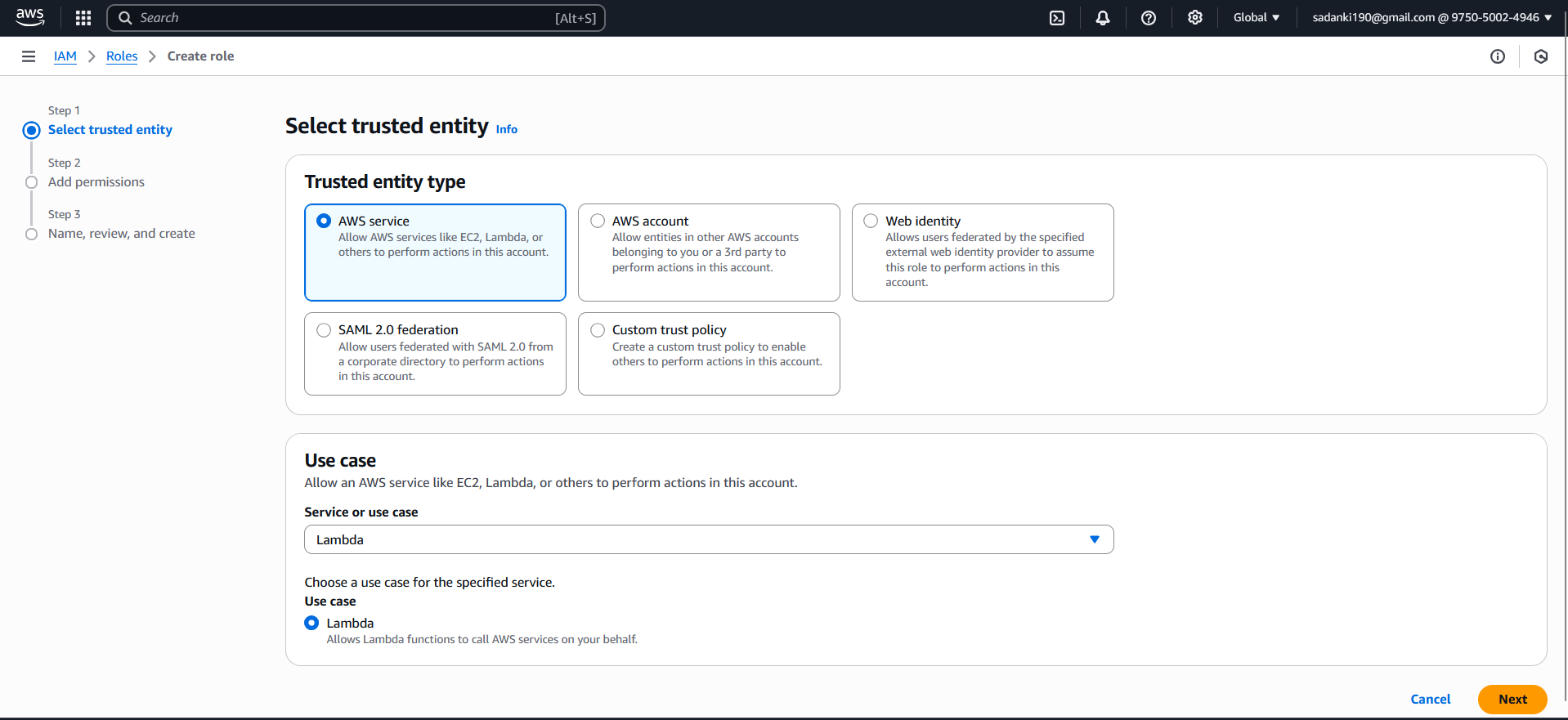
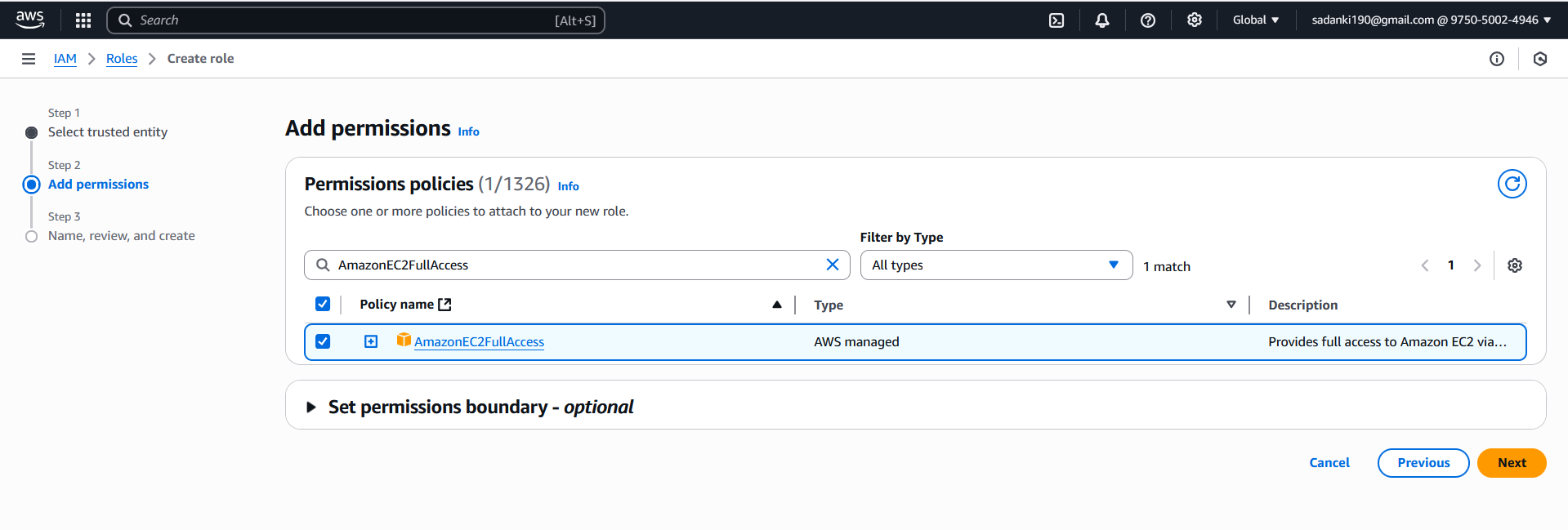
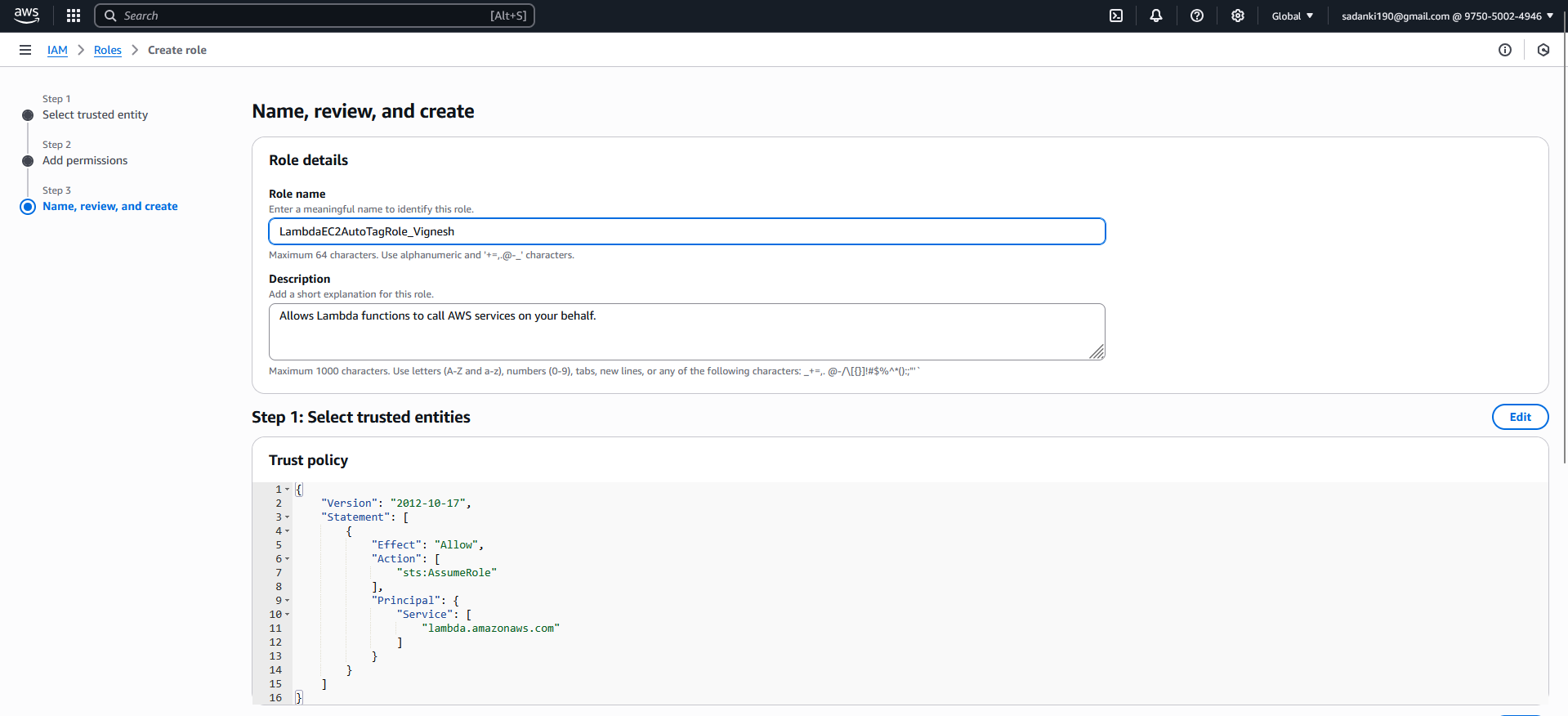
Assignment 5: Auto-Tagging EC2 Instances on Launch Using AWS Lambda and Boto3

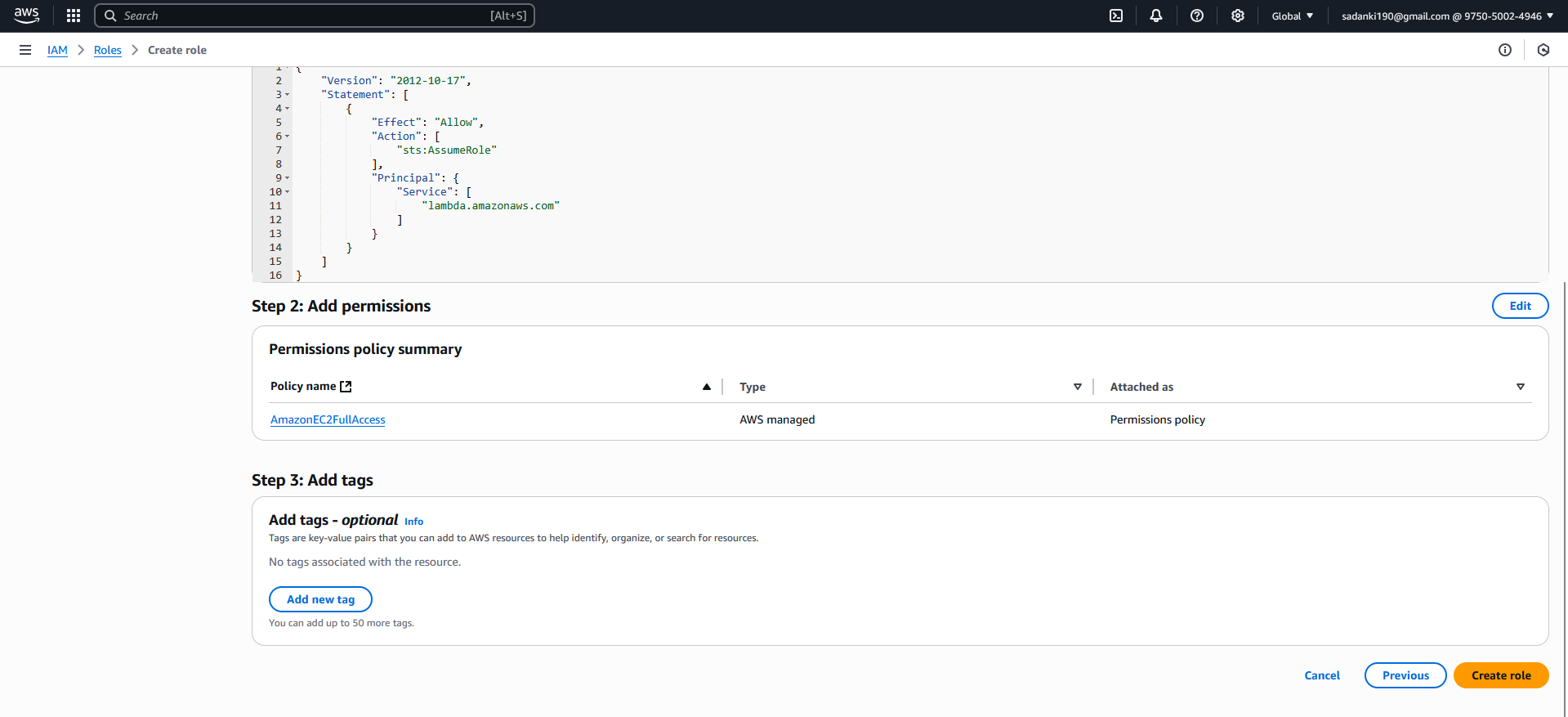
**Step 1: Create IAM Role for Lambda**

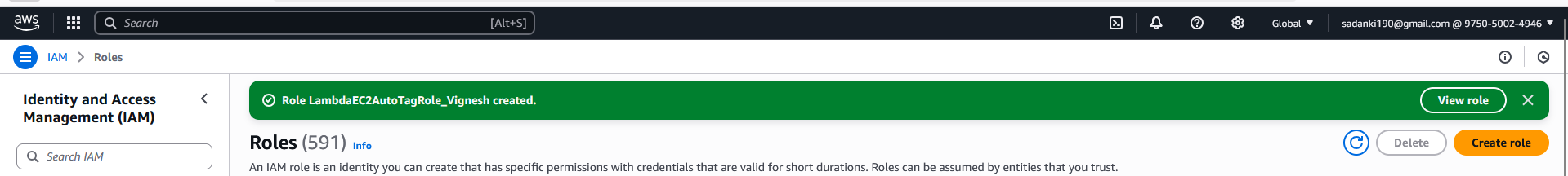
1. Go to the **AWS Management Console** → **IAM** → **Roles** → **Create Role**.
2. Select **AWS Service** → Choose **Lambda** as the use case.
3. Click **Next: Permissions**.
4. Search for and **attach the policy**: AmazonEC2FullAccess.
5. Click **Next: Tags** (skip adding tags).
6. Click **Next: Review**.
7. Name the role, e.g., LambdaEC2AutoTagRole.
8. Create the role.





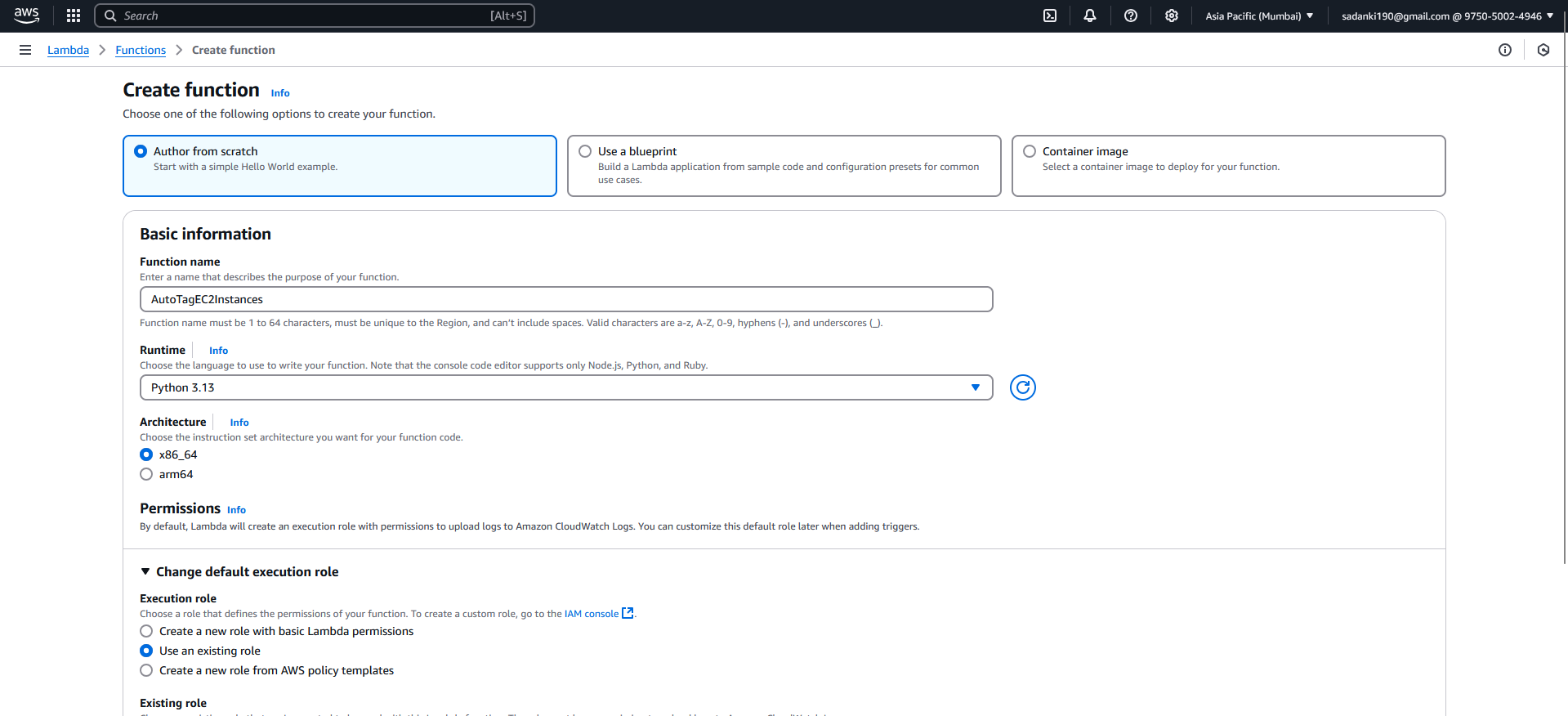


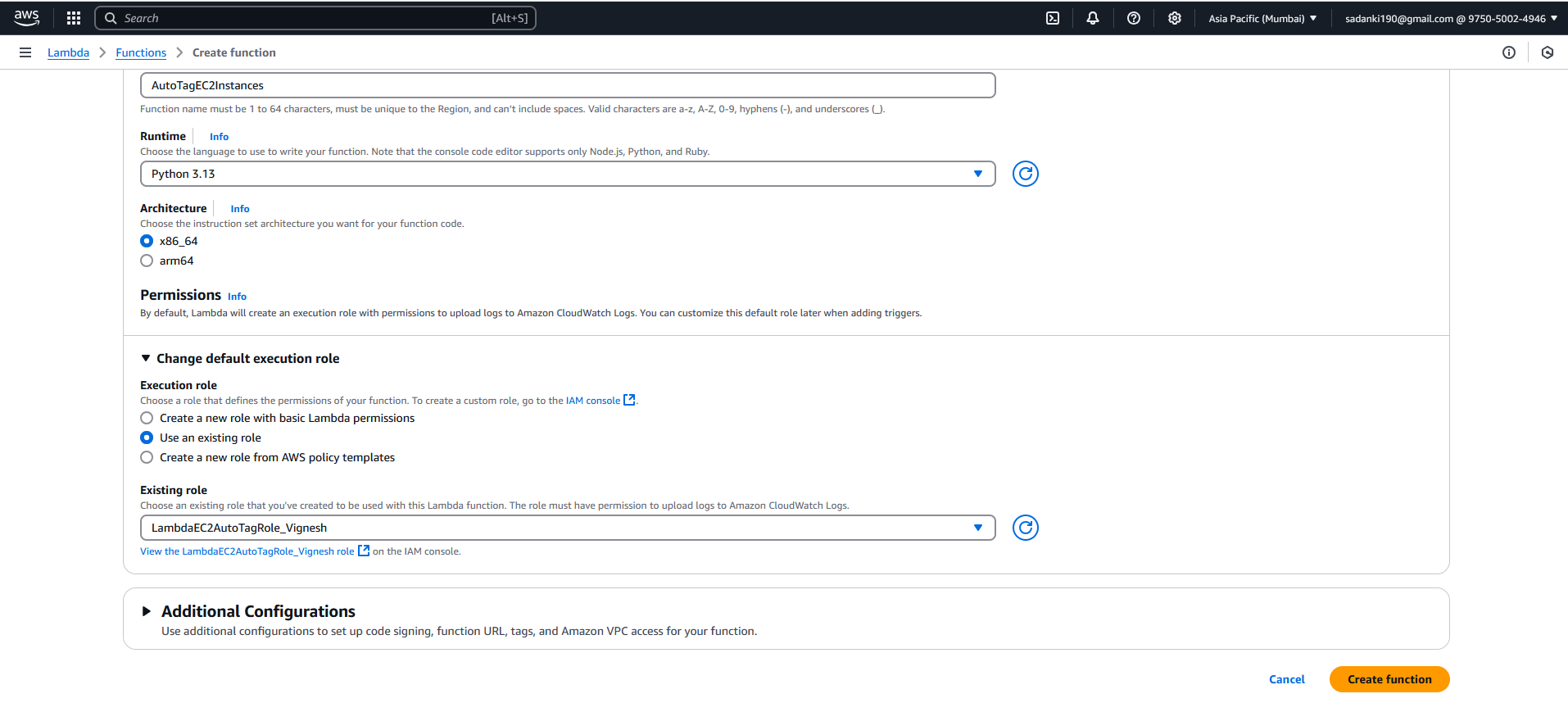




**Step 2: Create the Lambda Function**

1. Go to **AWS Lambda Console** → **Create function**.
2. Choose **Author from scratch**.
3. Function name: AutoTagEC2Instances.
4. Runtime: **Python 3.9** (or any 3.x).
5. Under **Permissions**, choose **Use an existing role**.
6. Select the IAM role created in Step 2 (LambdaEC2AutoTagRole).
7. Click **Create function**.



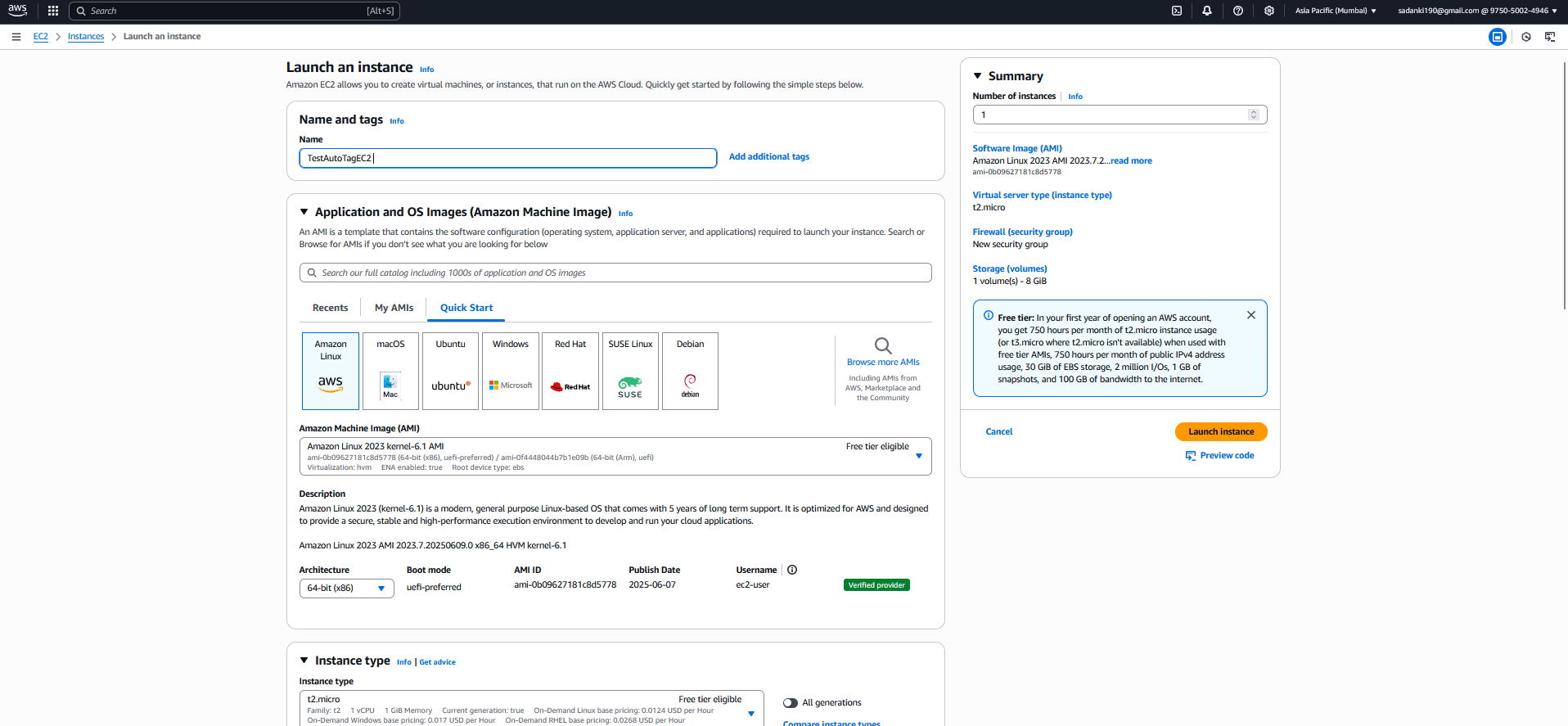


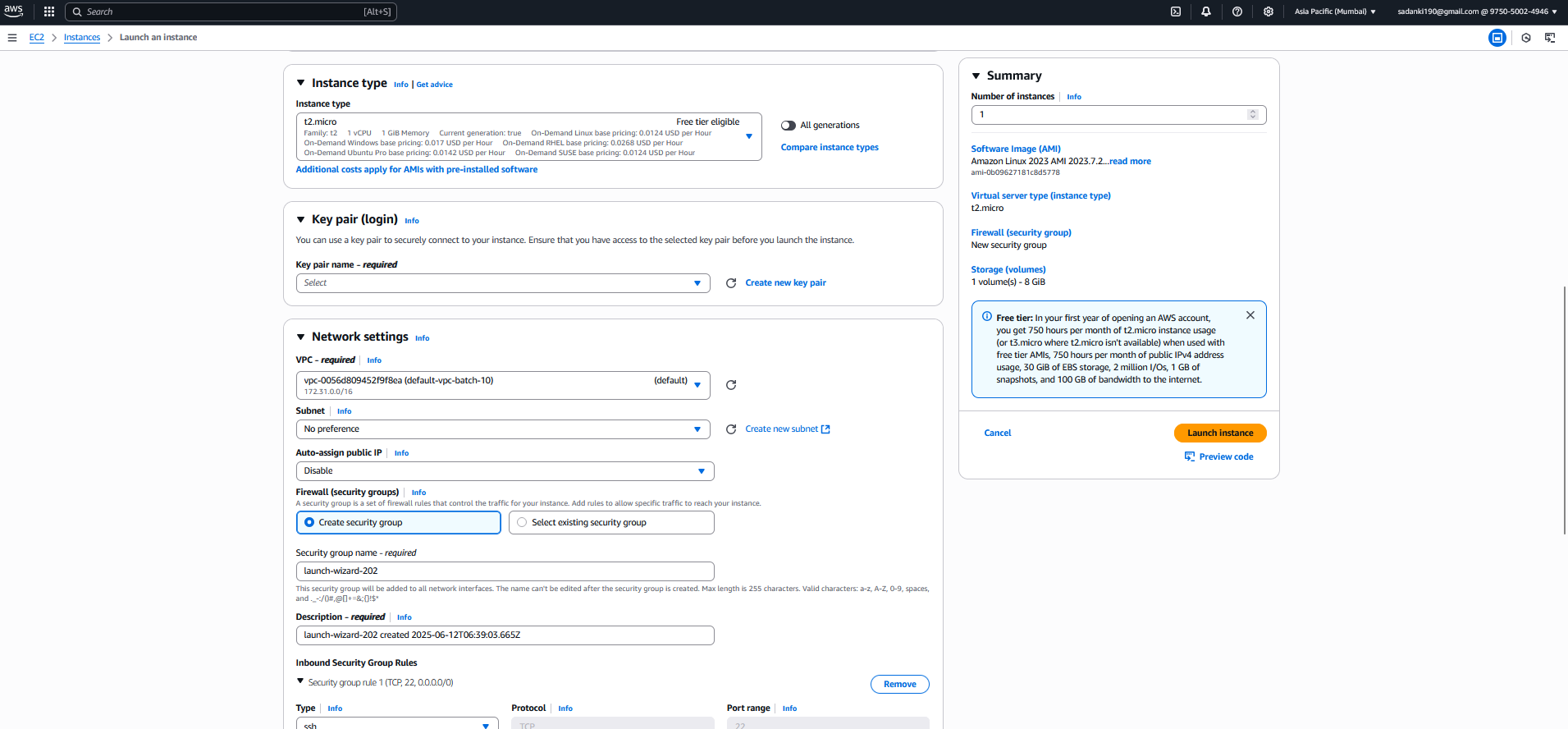
**Step 2: Launch a New EC2 Instance**

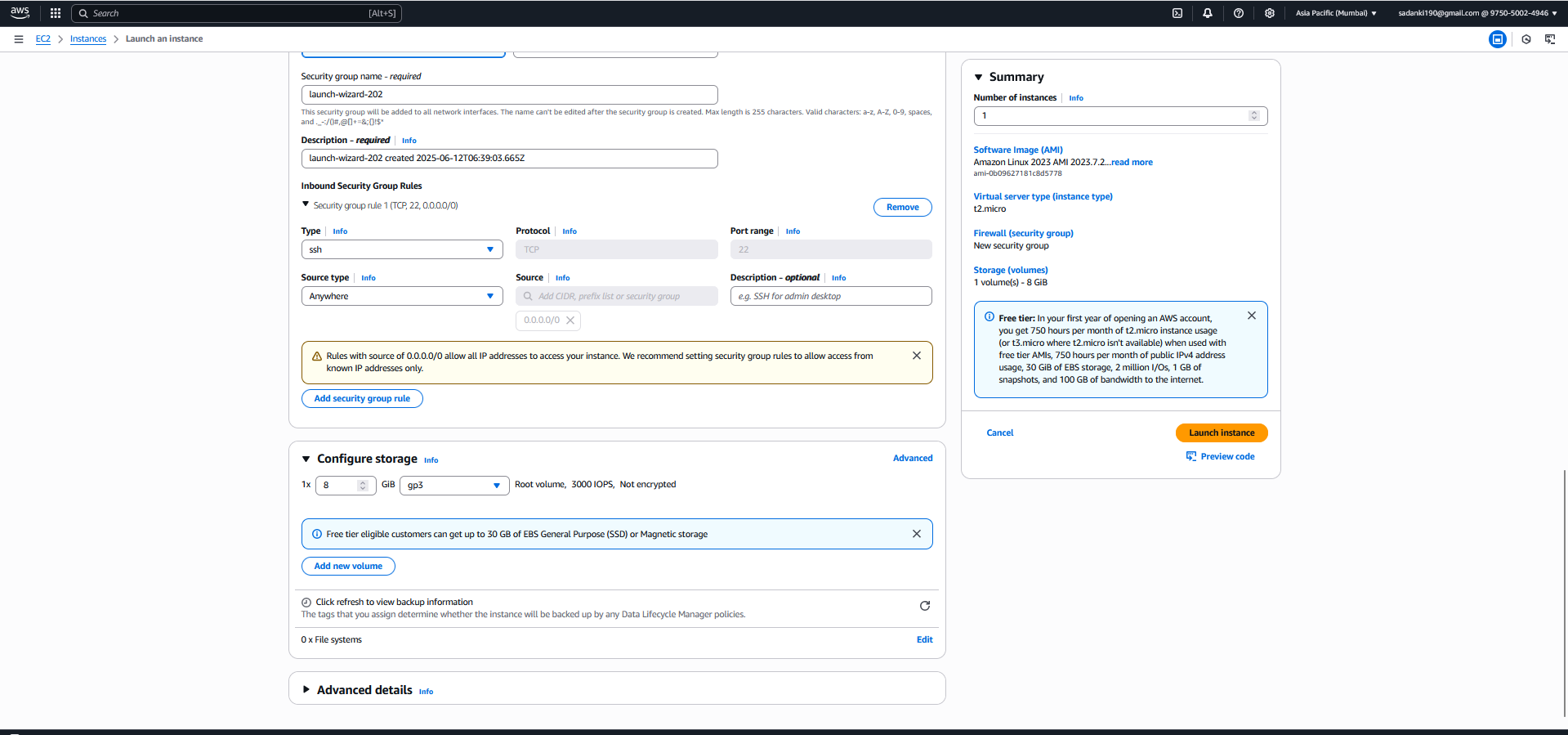
1. Go to the [EC2 Dashboard](https://console.aws.amazon.com/ec2/)
2. Click **Launch Instance**
3. **Name and Tags** (optional):
   * Name: TestAutoTagEC2 (for clarity)
   * You don’t need to add any manual tags — Lambda will add them
4. **Application and OS Images (AMI)**:
   * Choose: ✅ **Amazon Linux 2 AMI**
5. **Instance Type**:
   * ✅ Choose t2.micro (Free tier eligible)
6. **Key Pair**:
   * You can choose **“Proceed without key pair”** (for test only)
7. **Network Settings**:
   * Accept defaults (or uncheck “auto-assign public IP” if you're just testing)
8. **Configure Storage**:
   * Leave default (8 GiB gp2)
9. **Advanced Details**:
   * Leave everything as is (no need to set user data)
10. Click **Launch Instance**

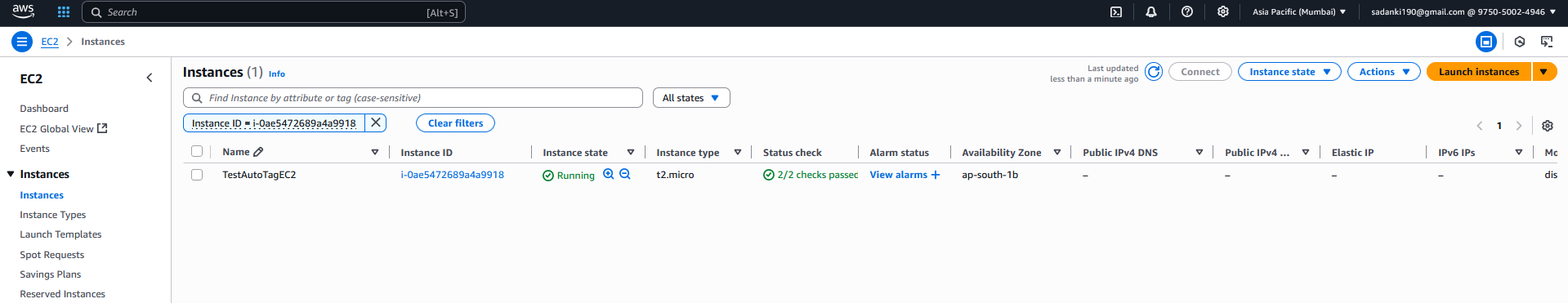
**Launch an EC2 Instance (Real Test)**

1. Go to **EC2 → Instances → Launch Instance**
2. Choose:
   * AMI: Amazon Linux 2
   * Instance Type: t2.micro (free-tier)
   * Keep default settings
3. Launch the instance
4. Wait 1–2 minutes
5. Go to **EC2 → Instances → Tags** tab
6. ✅ You should see:
   * LaunchDate = today’s date
   * Environment = Development









Step 4: Write the Lambda Code:

import boto3

import datetime

def lambda\_handler(event, context):

print("Received event:", event)

ec2 = boto3.client('ec2')

try:

instance\_id = event['detail']['instance-id']

current\_date = datetime.datetime.utcnow().strftime('%Y-%m-%d')

tags = [

{'Key': 'LaunchDate', 'Value': current\_date},

{'Key': 'Environment', 'Value': 'Development'}

]

ec2.create\_tags(Resources=[instance\_id], Tags=tags)

print(f"Tagged instance {instance\_id} with: {tags}")

return {

'statusCode': 200,

'body': f"Instance {instance\_id} tagged successfully."

}

except KeyError as e:

print(f"Missing expected event key: {str(e)}")

return {

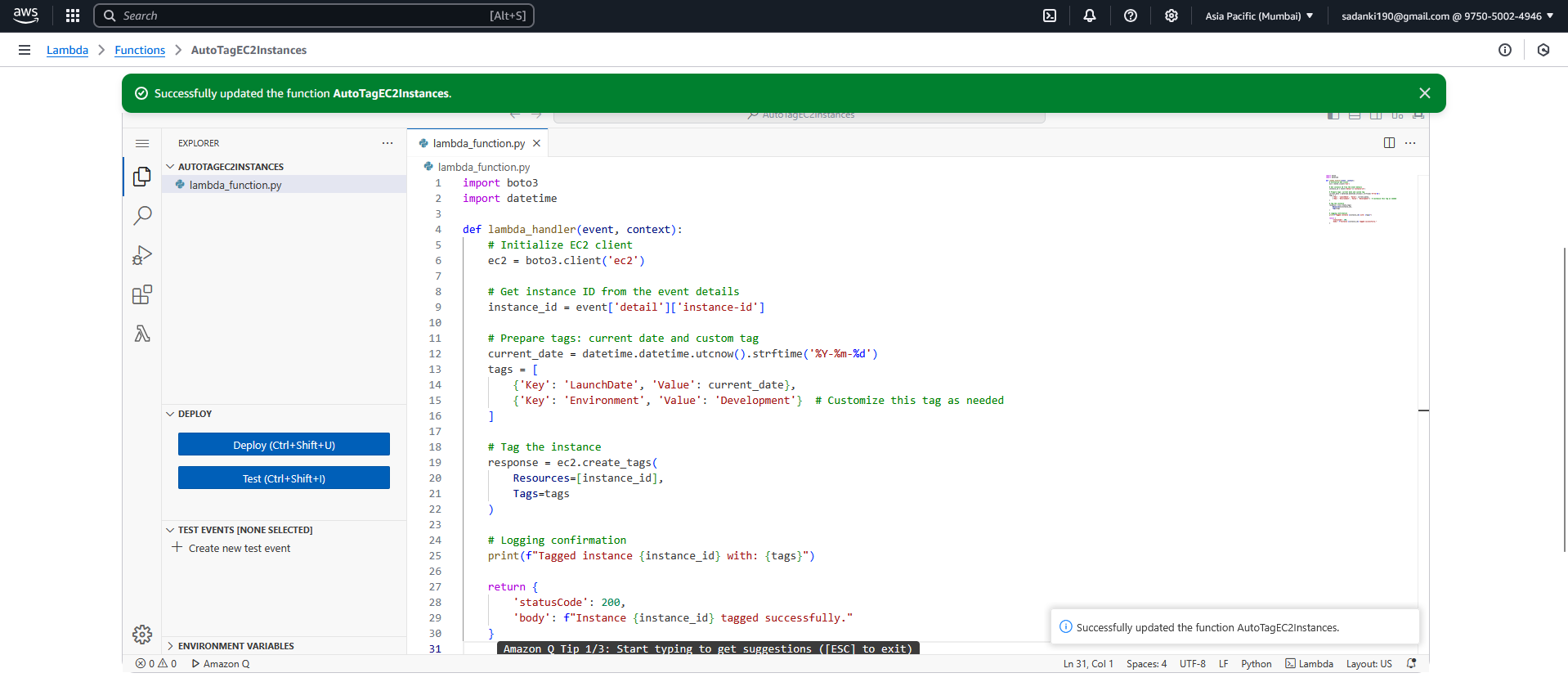
'statusCode': 400,

'body': f"Event missing key: {str(e)}"

}

This script:

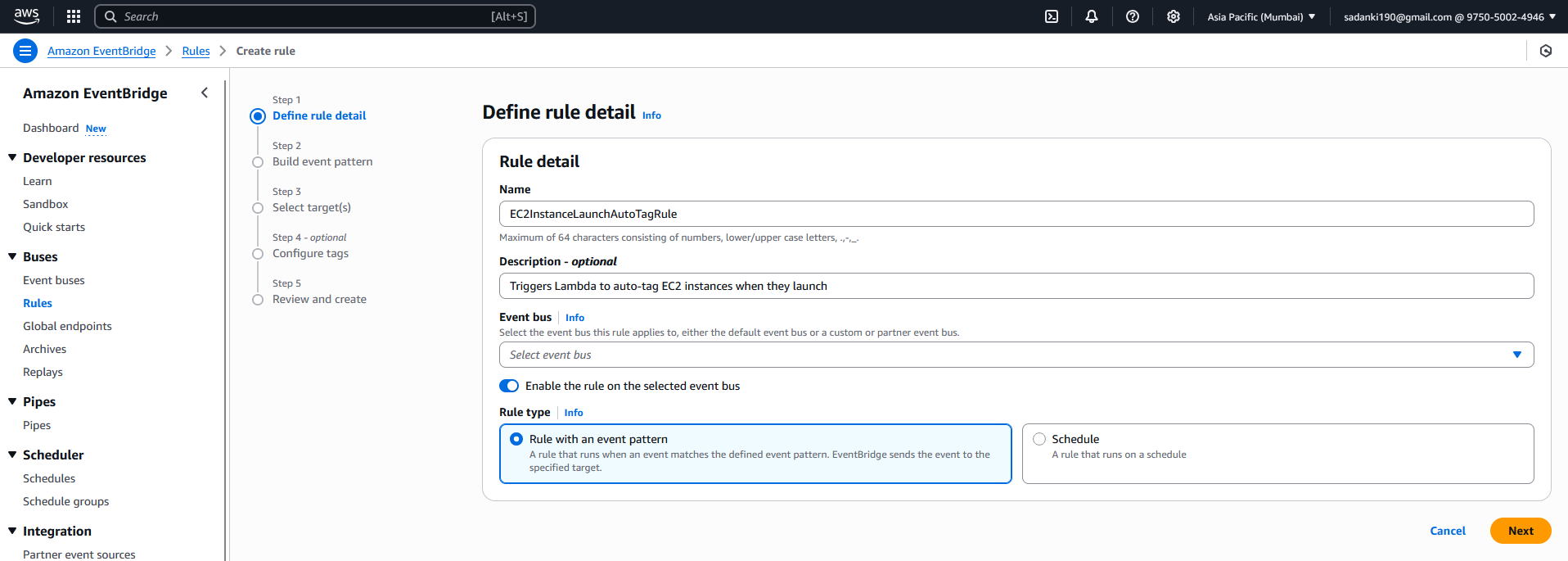
* Extracts the instance ID from the CloudWatch event.
* Tags the instance with current date and "Environment: Development".
* Prints confirmation in CloudWatch Logs.



Step 4: **Creating CloudWatch Event Rule for Auto-Tagging EC2 Instances** (EventBridge)

Step A: Define Rule Detail

* Name:  
  Choose a unique name for your rule, max 64 characters.  
  Example: EC2InstanceLaunchAutoTagRule
* Description (optional):  
  Example: Triggers Lambda to auto-tag EC2 instances when they launch
* Event bus:  
  Select default (unless you have a custom event bus).
* Enable the rule:  
  Leave checked (so the rule is active immediately).
* Rule type:  
  Choose Rule with an event pattern (this means the rule triggers when specific events happen).



**Step B: Build Event Pattern**

**1. Event source:**

Select:  
✅ **AWS events or EventBridge partner events**

**2. Creation method:**

Select:  
✅ **Use pattern form** (recommended)

This allows you to pick the service and event type using drop-downs.

**3. AWS service:**

Choose:  
✅ **EC2**

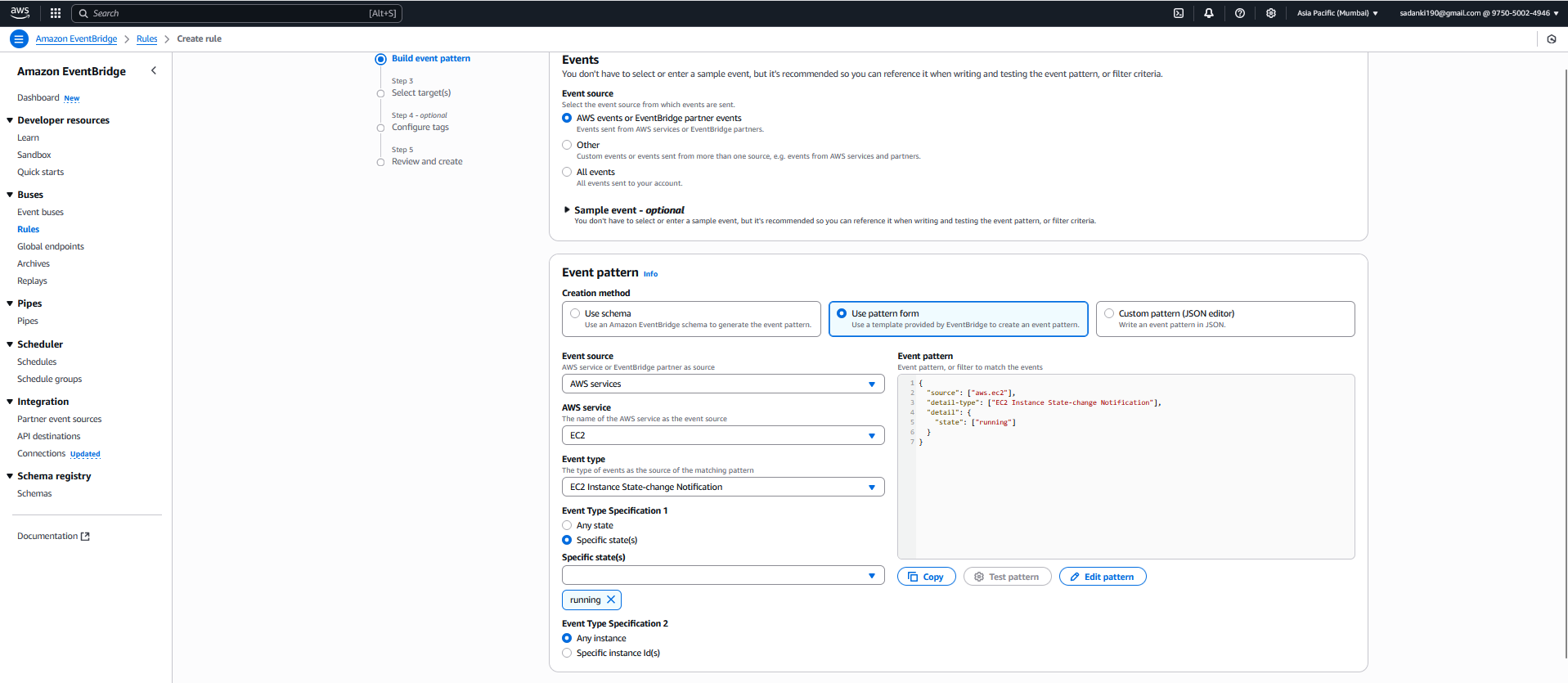
**4. Event type:**

Choose:  
✅ **EC2 Instance State-change Notification**

**5. Specific event detail (state):**

* You will now see a drop-down for instance state.
* From the list, select:  
  ✅ **running**

This ensures the rule triggers when an EC2 instance changes to the running state (i.e., when launched)



**Step C: Select Target(s)**

**1. Target types:**

**Select:  
✅ AWS service**

**2. Select target:**

**From the dropdown that appears:  
✅ Lambda function**

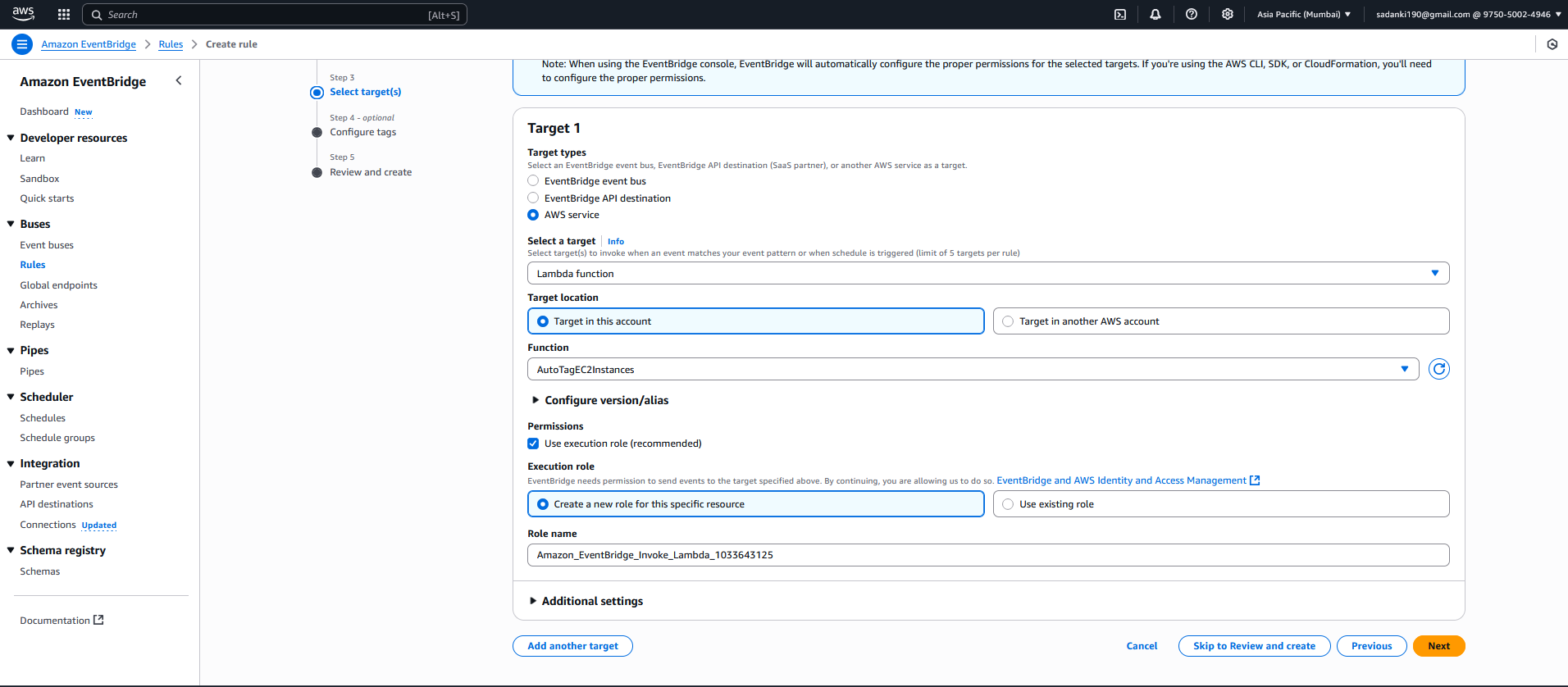
**3. Function:**

* **Once you select Lambda function, another dropdown will appear.**
* **Select your Lambda function:  
  ✅ AutoTagEC2Instances (or whatever name you gave your function)**

**4. Additional settings:**

**You can leave all defaults here unless you have advanced requirements.**

* **No need to configure constant JSON, input transformer, etc.**
* **Leave "Retry policy", "Dead-letter queue", and "Enable trigger" settings as is.**

****

**✅ Permissions Note:**

**You’ll see a note:**

***EventBridge will automatically configure the proper permissions for the selected targets.***

**✅ Just proceed — AWS will automatically give EventBridge permission to invoke your Lambda function.**

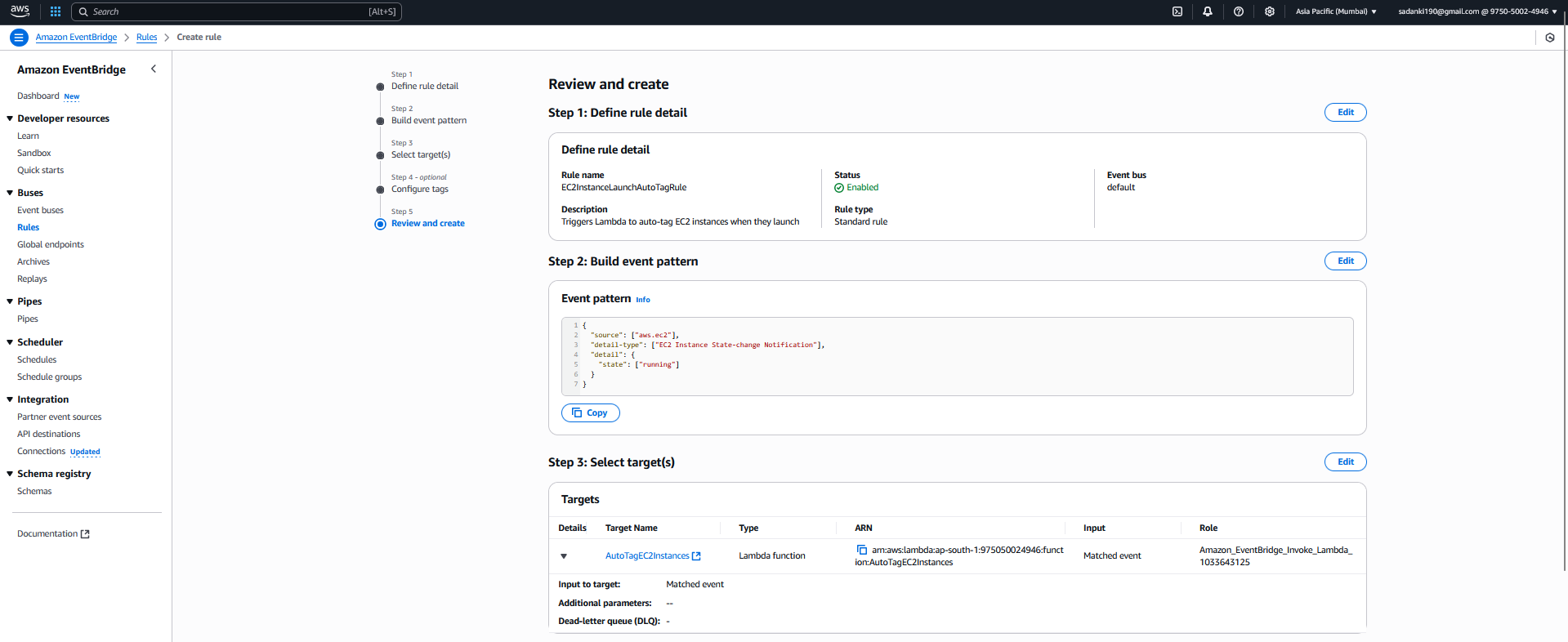
**Step D (Optional): Configure Tags**

* Add tags to your EventBridge rule if you want for management or billing purposes.
* This is optional; you can skip it for now.

**Step E: Review and Create**

* Review all the details:
  + Rule Name
  + Event bus: default
  + Event pattern (EC2 running state)
  + Target Lambda function
* Click **Create rule**.

Once the rule is created, it will listen for any EC2 instance state changes to **running** and trigger your Lambda function, which auto-tags the instance.



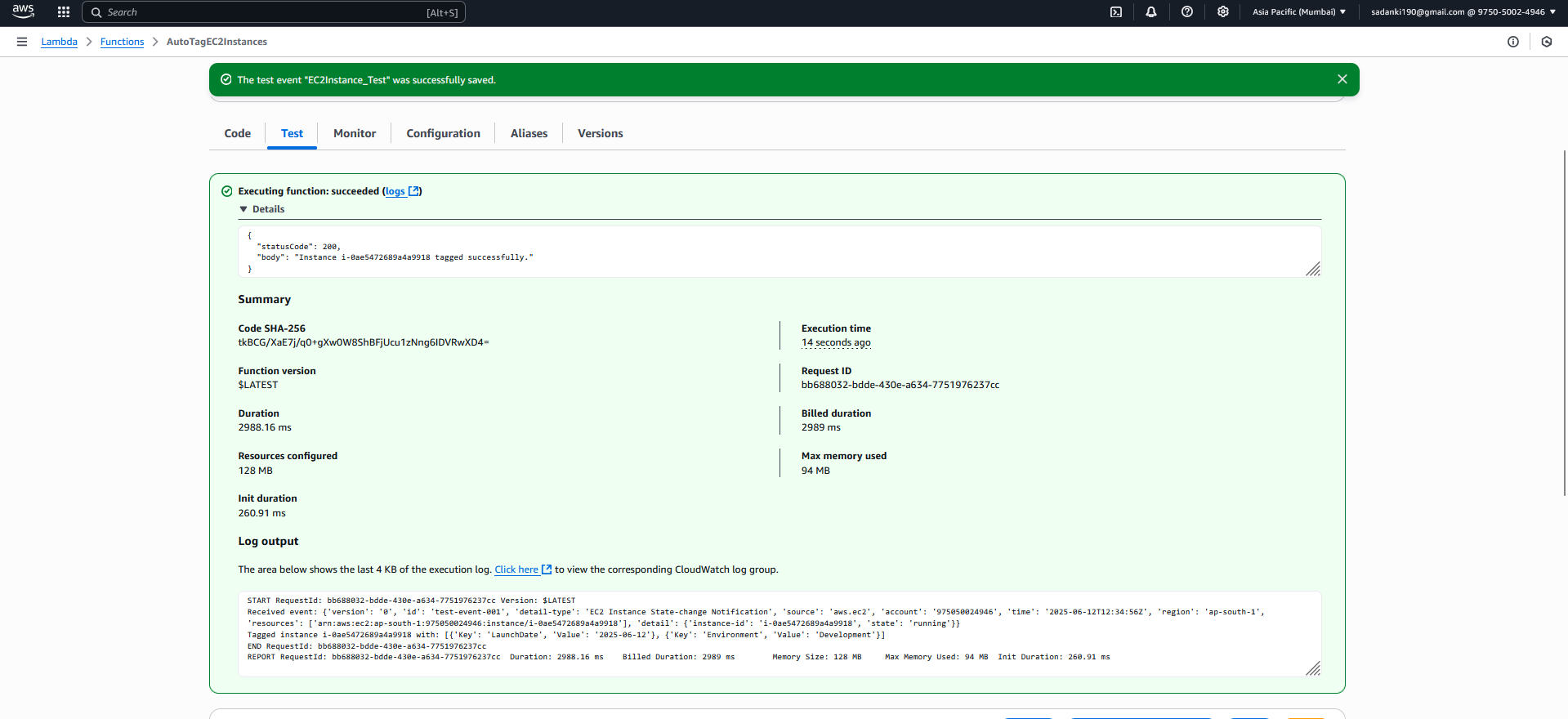
**Step 5: Test the Setup**

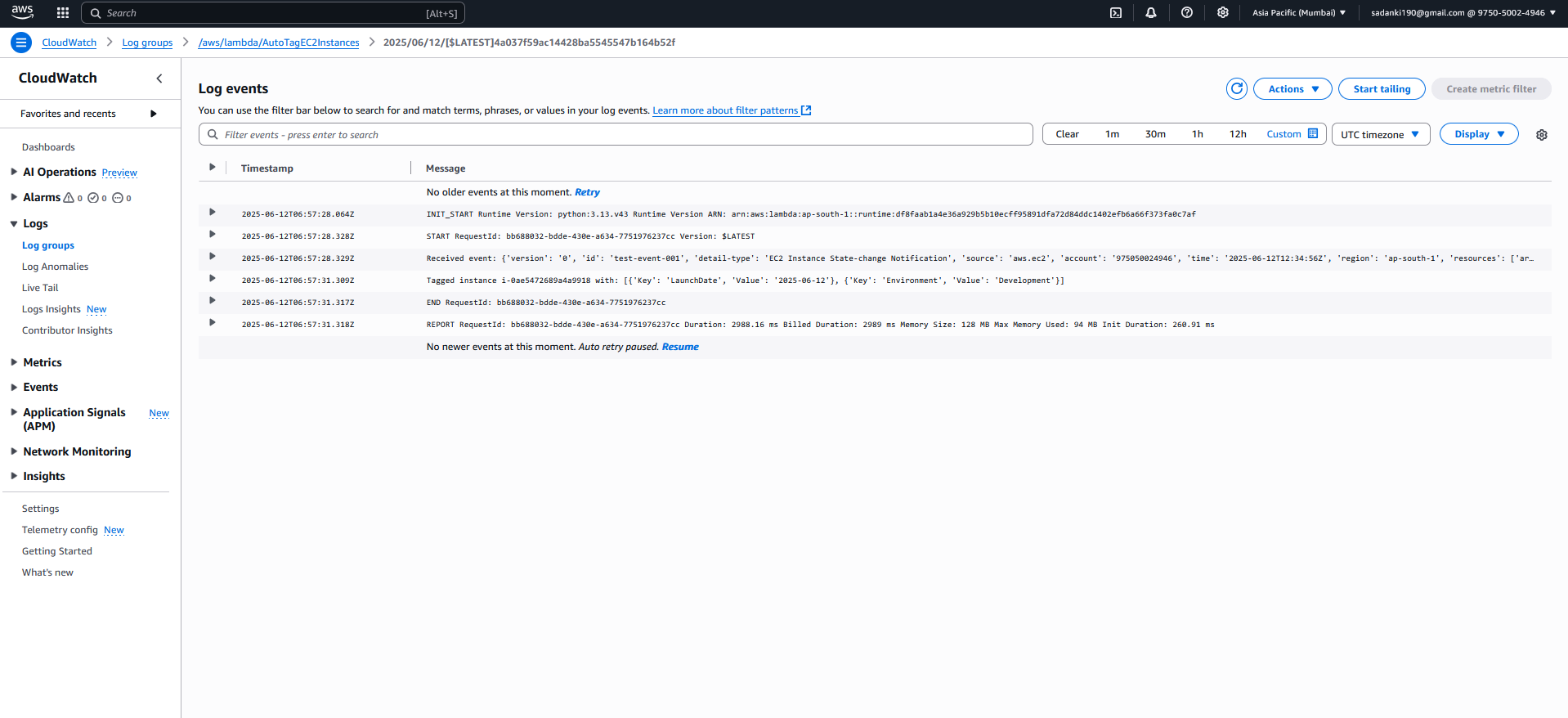
1. Go to **EC2 Dashboard** → **Launch Instance**.
2. Launch a free-tier eligible instance (e.g., Amazon Linux 2, t2.micro).
3. Wait 1–2 minutes for the instance to fully start.
4. Go to **EC2 Instances** → Select your new instance.
5. Check the **Tags** tab.
6. You should see:
   * LaunchDate tag with the current date.
   * Environment tag with "Development".

**Notes**

* Your Lambda function logs (confirmation prints) can be checked in **CloudWatch Logs** under /aws/lambda/AutoTagEC2Instances.
* Make sure your Lambda execution role has the AmazonEC2FullAccess policy.
* Use **free-tier regions** and **free-tier instance types** to avoid charges.

Testing:





log output confirms that the Lambda function worked perfectly and completed **Assignment 5** as intended. Here's the breakdown:

What Happened ?

| **✅ Step** | **🔍 Confirmation from Your Log** |
| --- | --- |
| Lambda triggered | START RequestId: ... |
| Received correct event | 'detail': {'instance-id': 'i-0ae5472689a4a9918', 'state': 'running'} |
| Auto-tagging succeeded | Tagged instance i-0ae5472689a4a9918 with: [{'Key': 'LaunchDate', 'Value': '2025-06-12'}, {'Key': 'Environment', 'Value': 'Development'}] |
| Execution completed | statusCode: 200 & "Instance i-0ae5472689a4a9918 tagged successfully." |

Now What Should You Do?

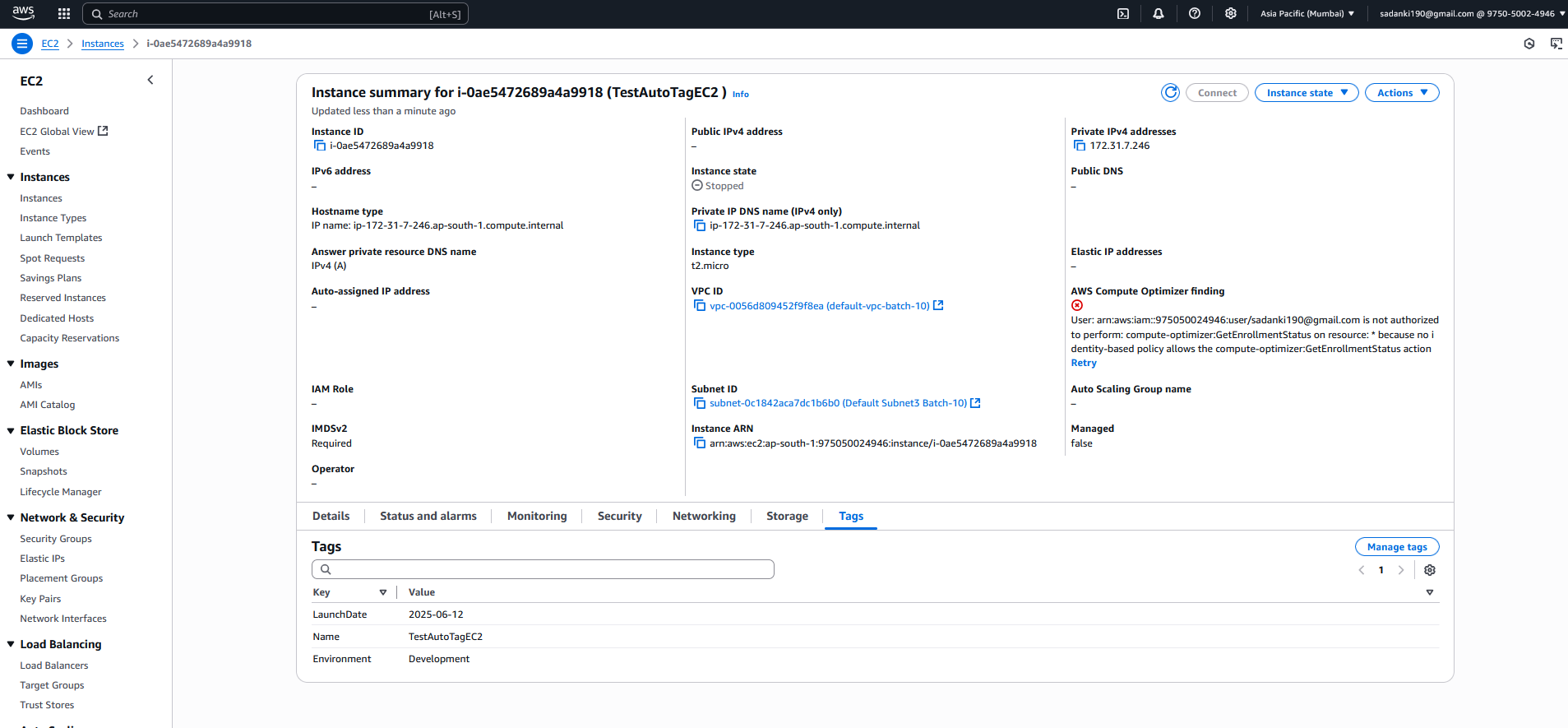
**1. Check EC2 Console**

Go to:

* **EC2 → Instances → i-0ae5472689a4a9918**
* **Tags tab**

You should see:

| **Key** | **Value** |
| --- | --- |
| LaunchDate | 2025-06-12 |
| Environment | Development |



**Assignment 5 Summary**.

**📘 Assignment 5: Auto-Tagging EC2 Instances on Launch Using AWS Lambda and Boto3**

**🎯 Objective**

To automate the tagging of newly launched EC2 instances with:

* The **current date** (LaunchDate)
* A custom tag (e.g., Environment = Development)

This helps in maintaining consistent resource labeling, cost tracking, and governance.

**🛠️ What Was Implemented**

**✅ 1. IAM Role Setup**

* Role name: LambdaEC2AutoTagRole\_Vignesh
* Permissions attached:
  + AmazonEC2FullAccess – for tagging EC2 instances
  + AWSLambdaBasicExecutionRole – for logging in CloudWatch

**✅ 2. Lambda Function**

* Name: AutoTagEC2Instances
* Runtime: Python 3.x
* Function Logic:
  + Extract instance ID from the event payload
  + Tag the instance with:
    - LaunchDate: current UTC date
    - Environment: Development

**✔️ Sample Code Snippet:**

import boto3

import datetime

def lambda\_handler(event, context):

ec2 = boto3.client('ec2')

instance\_id = event['detail']['instance-id']

current\_date = datetime.datetime.utcnow().strftime('%Y-%m-%d')

tags = [

{'Key': 'LaunchDate', 'Value': current\_date},

{'Key': 'Environment', 'Value': 'Development'}

]

ec2.create\_tags(Resources=[instance\_id], Tags=tags)

print(f"Tagged instance {instance\_id} with: {tags}")

return {

'statusCode': 200,

'body': f"Instance {instance\_id} tagged successfully."

}

**✅ 3. EventBridge Rule (CloudWatch)**

* Rule name: EC2InstanceLaunchAutoTagRule
* Trigger: EC2 **state = running**
* Target: Lambda function AutoTagEC2Instances
* Effect: Lambda is invoked **immediately after EC2 is launched**

**✅ 4. Testing**

* Launched an EC2 instance: TestAutoTagEC2
* Used **manual test event** with correct JSON payload
* Verified CloudWatch logs:
* Tagged instance i-0ae5472689a4a9918 with: [{'Key': 'LaunchDate', 'Value': '2025-06-12'}, {'Key': 'Environment', 'Value': 'Development'}]
* ✅ Tags successfully applied to the instance

**✅ Result**

* EC2 instances are now auto-tagged immediately upon launch.
* Manual testing and logs confirmed tagging worked as expected.

**👍 Benefits**

| **Feature** | **Benefit** |
| --- | --- |
| ✅ Automation | Eliminates manual tagging |
| ✅ Consistency | Ensures all instances follow tagging policy |
| ✅ Cost Tracking | Enables billing by tags |
| ✅ Security | Tracks who launched what, when, and why |

**⚠️ Limitations**

| **Limitation** | **Description** |
| --- | --- |
| ⏱️ Trigger Delay | Small delay before tags appear |
| 🧪 Region-Specific Setup | Rule and Lambda must be in same region |
| ❌ No conditional tagging | Same tags applied to all instances |