Project

Topic: Automated Backup of S3 Buckets using AWS Lambda.

Why AWS Lambda:

AWS Lambda is a serverless compute service provided by Amazon Web Services (AWS) that allows you to run code without provisioning or managing servers. Here are some key reasons why AWS Lambda is widely used:

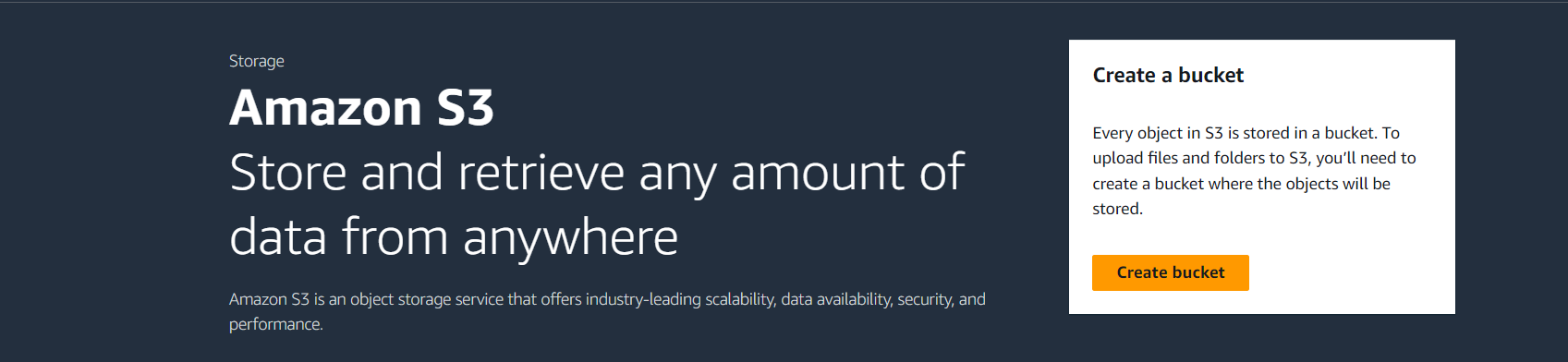
1. **Serverless Architecture**: Lambda abstracts away the underlying infrastructure, so you don't need to manage servers, operating systems, or scale provisioning.
2. **Pay-per-Use Pricing**: You pay only for the compute time you consume in increments of 100 milliseconds, and there is no charge when your code is not running.
3. **Automatic Scaling**: Lambda automatically scales your application by running code in response to each trigger (e.g., HTTP request, S3 event, DynamoDB update) individually.
4. **Integration with AWS Services**: Lambda integrates seamlessly with other AWS services like S3, DynamoDB, API Gateway, SNS, and more, allowing you to build powerful serverless applications.
5. **Support for Multiple Programming Languages**: Lambda supports multiple programming languages such as Python, Node.js, Java, Go, Ruby, and .NET Core, giving developers flexibility in choosing their preferred language.
6. **Event-Driven Programming Model**: Lambda functions are triggered by events, allowing you to build applications that respond to changes in data, user activity, or system states in real-time.
7. **Scalability and High Availability**: AWS Lambda manages the infrastructure and ensures high availability and fault tolerance. Your functions automatically scale out to handle increases in traffic.
8. **Easy Deployment and Management**: Lambda functions can be easily deployed and managed using AWS Management Console, AWS CLI, or SDKs. Versioning and aliasing allow you to deploy updates safely.
9. **Reduced Operational Overhead**: With Lambda, developers can focus more on writing code and less on managing infrastructure, reducing operational overhead and improving time-to-market.
10. **Cost Optimization**: Lambda's pay-per-use pricing model and automatic scaling help optimize costs by eliminating the need for over-provisioning and minimizing idle time.

Overall, AWS Lambda is popular because it simplifies the development and operations of applications, allows for rapid scaling and cost efficiency, and integrates seamlessly with the broader AWS ecosystem of services.

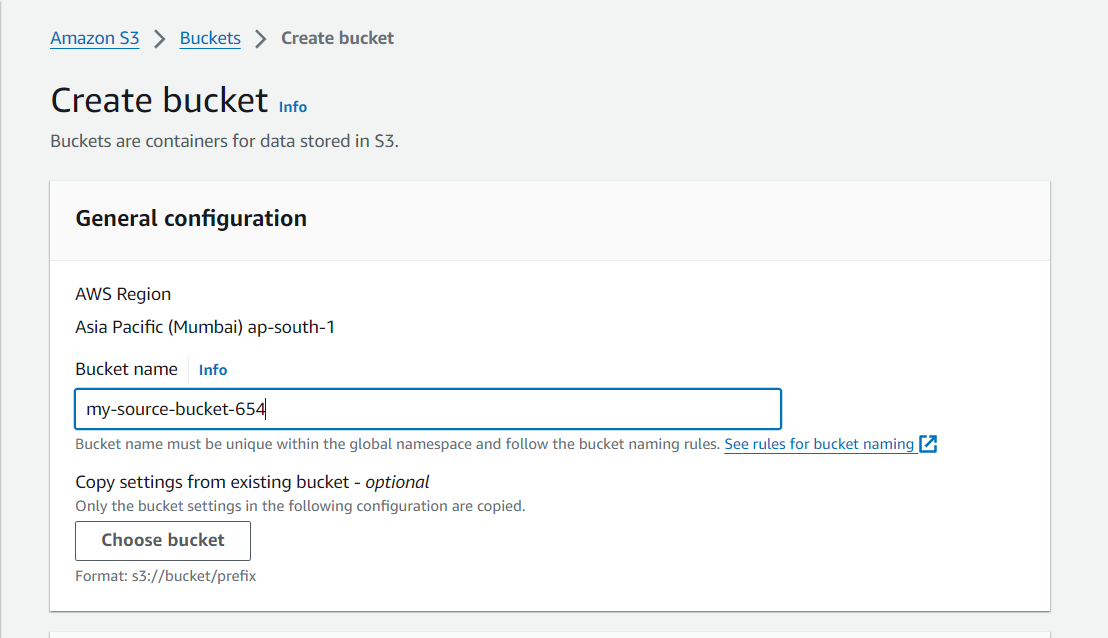
Steps:

**Step 1: Create S3 Buckets**

1. **Log in to the AWS Management Console**.
2. **Create the Source Bucket**:
   * Go to the S3 service.

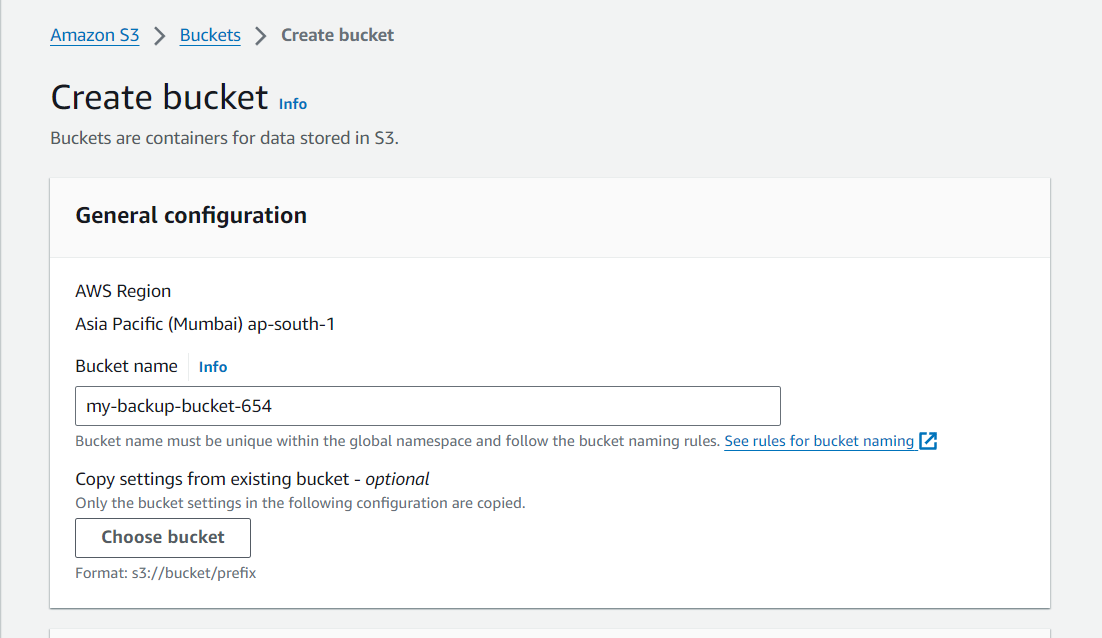


* + Click "Create bucket".



* + Enter a unique bucket name, e.g., my-source-bucket-654.
  + Select the region and leave other settings as default.
  + Click "Create bucket".

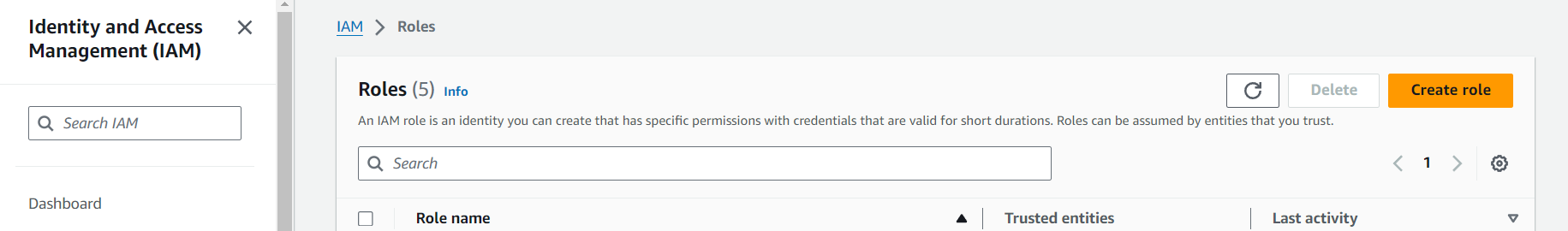
1. **Create the Backup Bucket**:



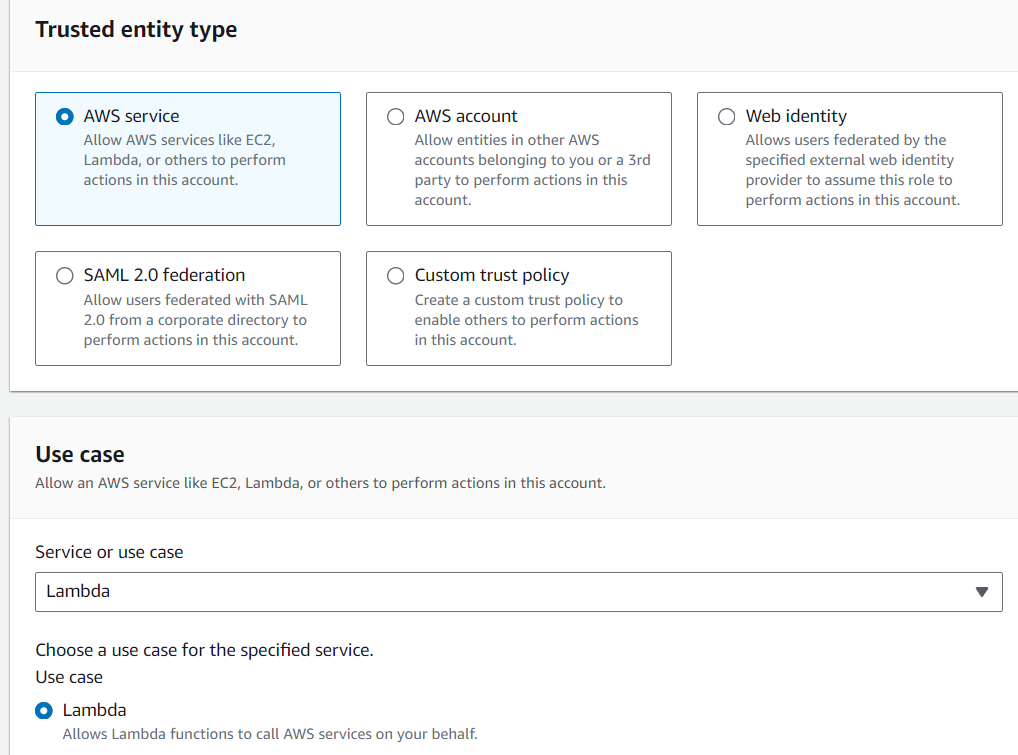
* + Repeat the same steps as above to create another bucket, e.g., my-backup-bucket-654.

**Step 2: Create IAM Role**

1. **Go to the IAM service**:
   * In the AWS Management Console, navigate to the IAM service.
2. **Create a New Role**:

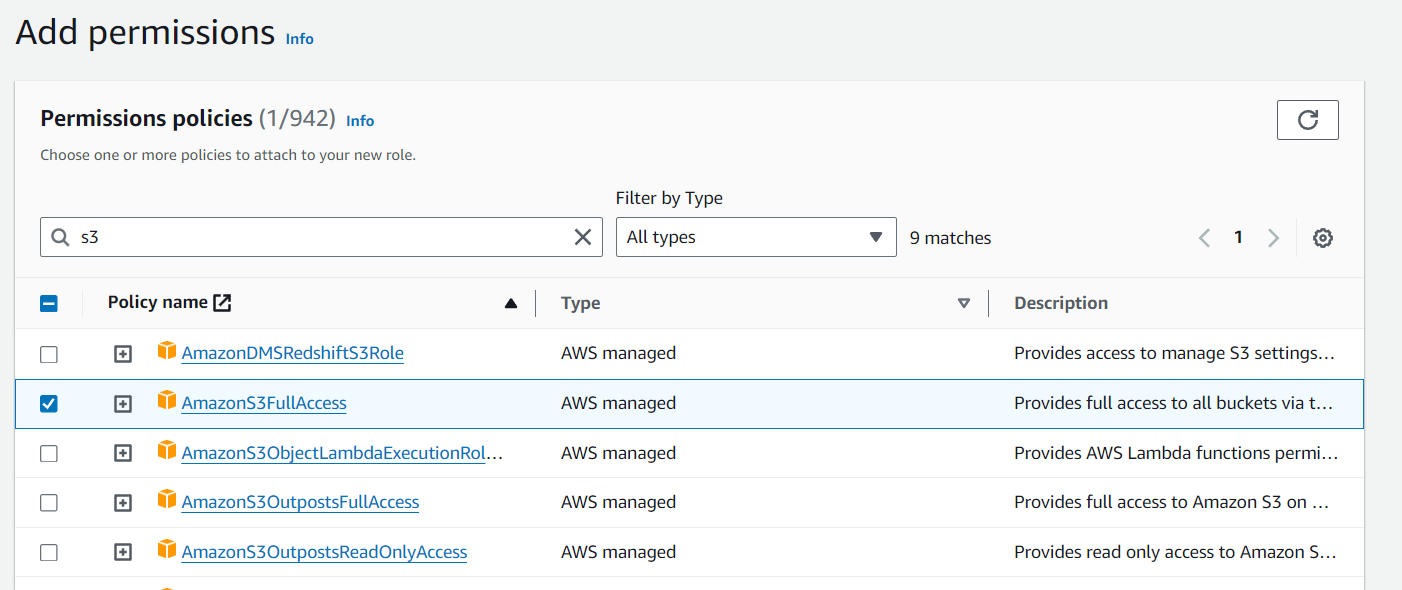


* + Click on "Roles" in the left menu.
  + Click "Create role".

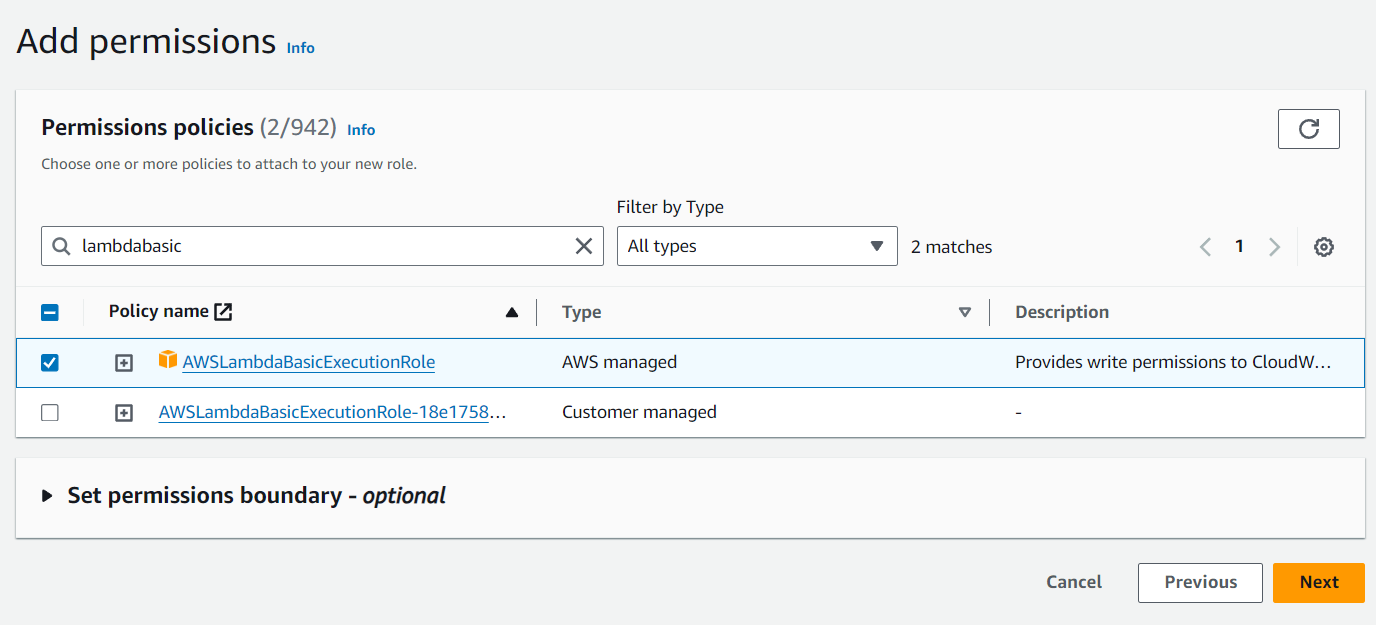


* + Select "AWS service" and then "Lambda".
  + Click "Next: Permissions".

1. **Attach Policies**:

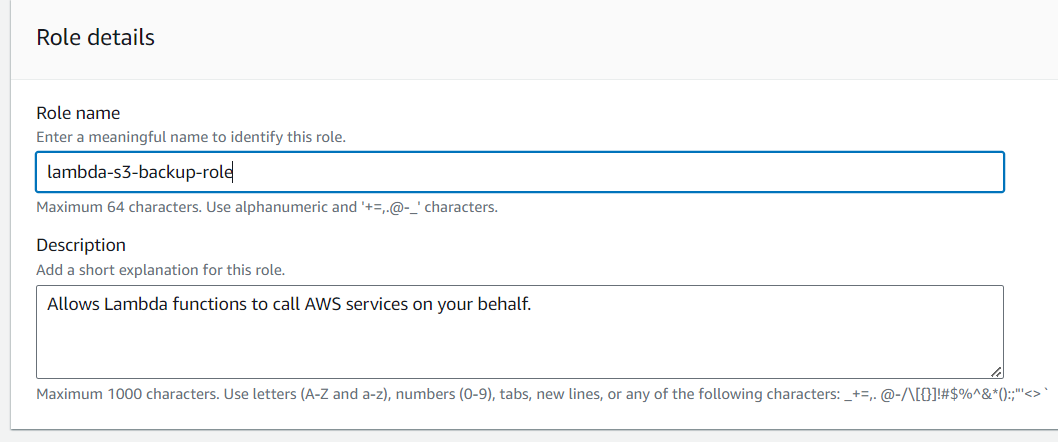


* + Search for and select the AmazonS3FullAccess policy.



* + Search for and select the AWSLambdaBasicExecutionRole policy.
  + Click "Next: Tags" (you can skip adding tags).
  + Click "Next: Review".

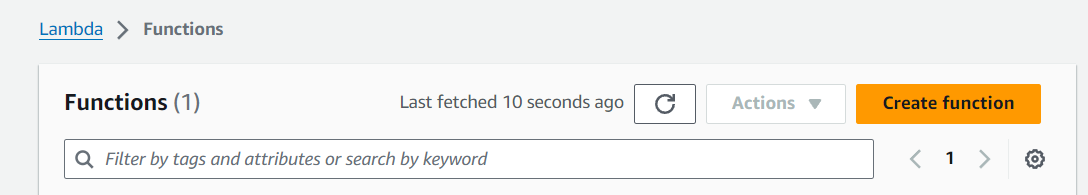
1. **Name the Role**:



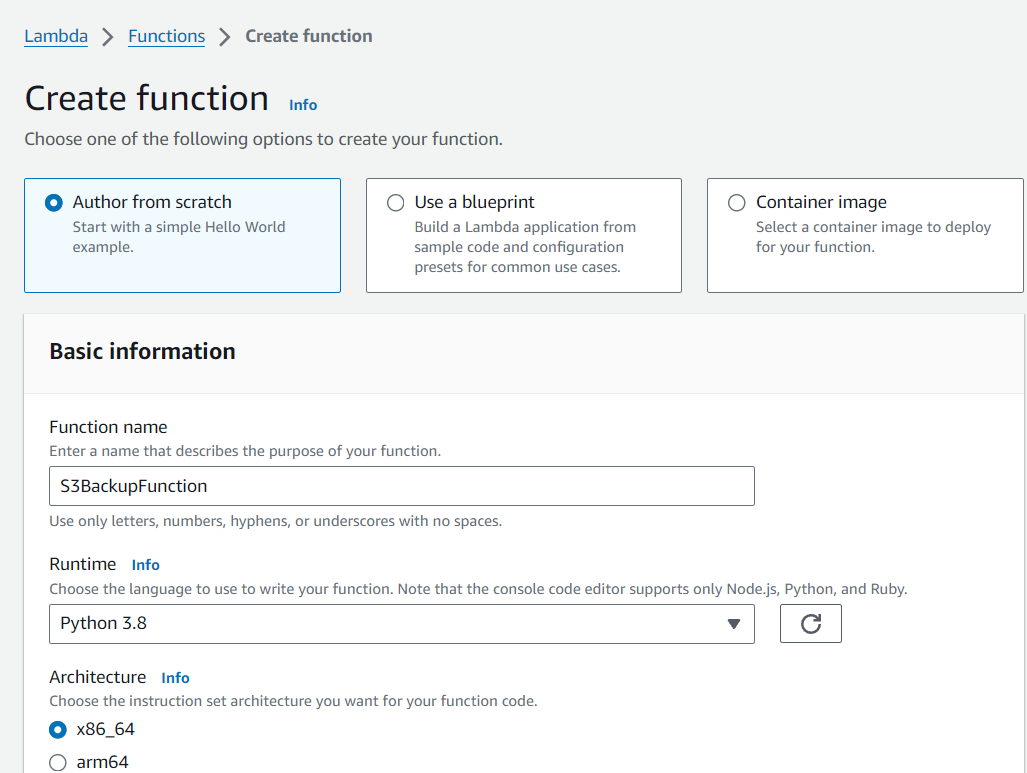
* + Enter a role name, e.g., lambda-s3-backup-role.
  + Click "Create role".

**Step 3: Create a Lambda Function**

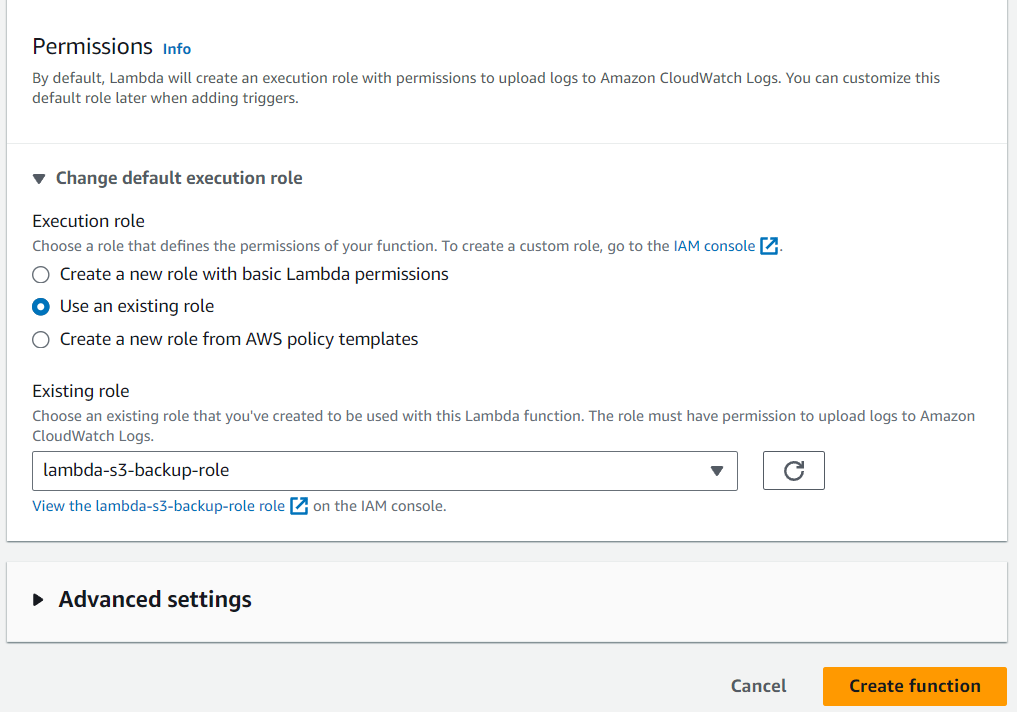
1. **Go to the Lambda service**:
   * In the AWS Management Console, navigate to the Lambda service.
2. **Create a New Function**:



* + Click "Create function".
  + Choose "Author from scratch".

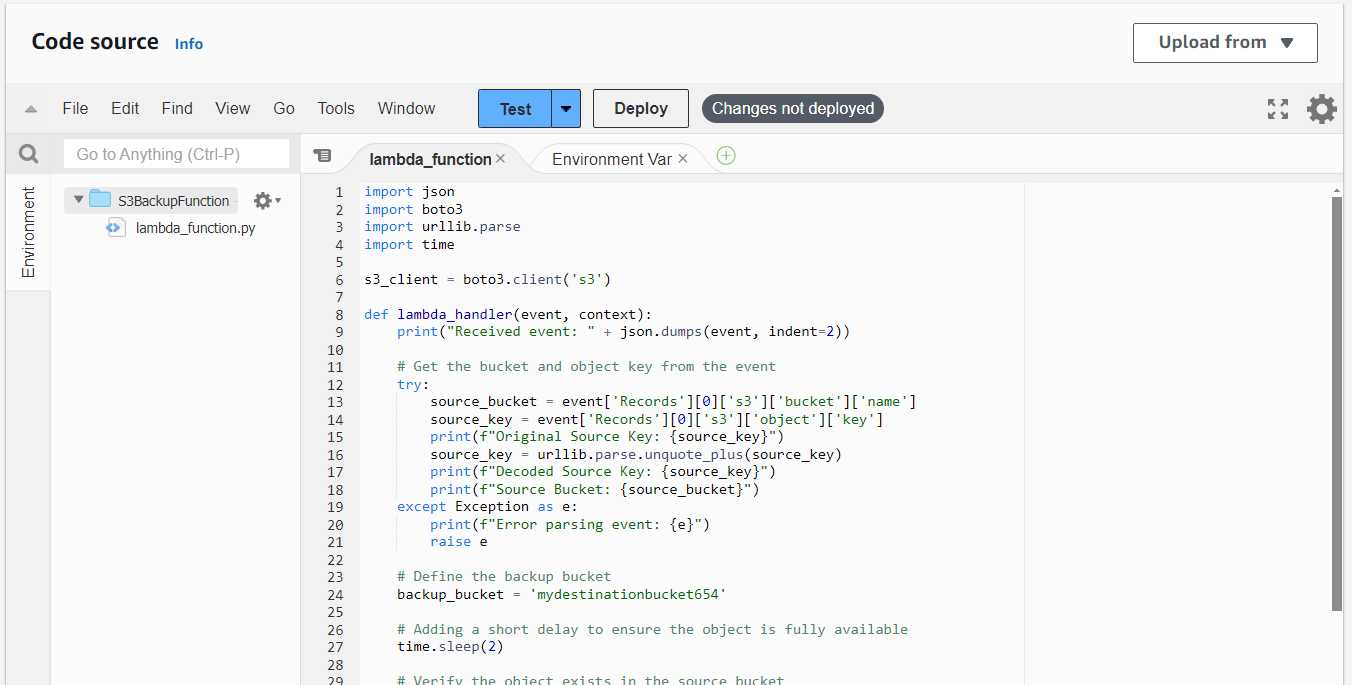


* + Enter a function name, e.g., S3BackupFunction.
  + Choose the runtime, e.g., Python 3.8.
  + Under "Permissions", choose "Use an existing role".



* + Select the role created in Step 2 (lambda-s3-backup-role).
  + Click "Create function".

1. **Add Function Code**:
   * In the function's code editor, replace any existing code with the following:



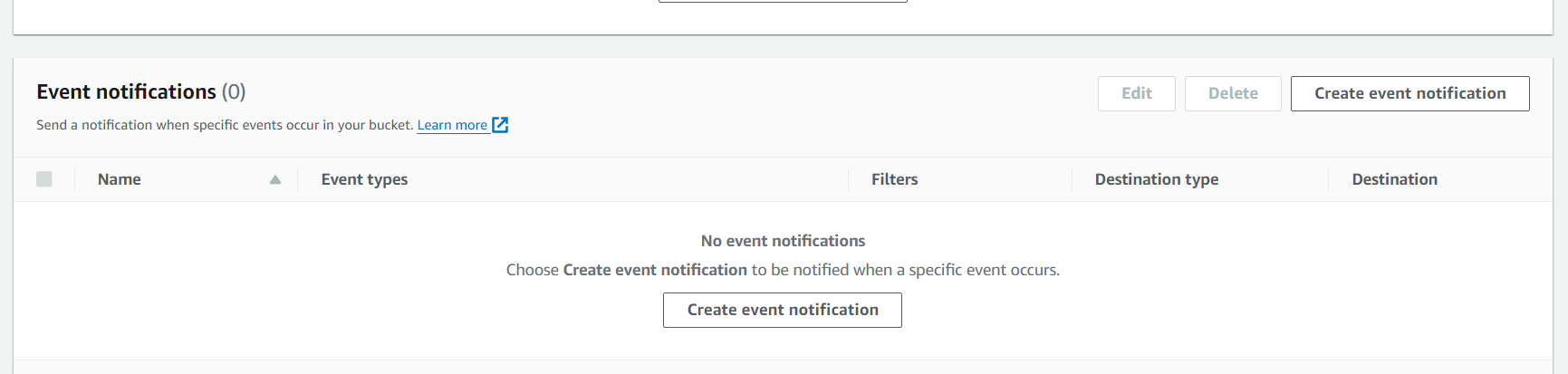
python

Copy code

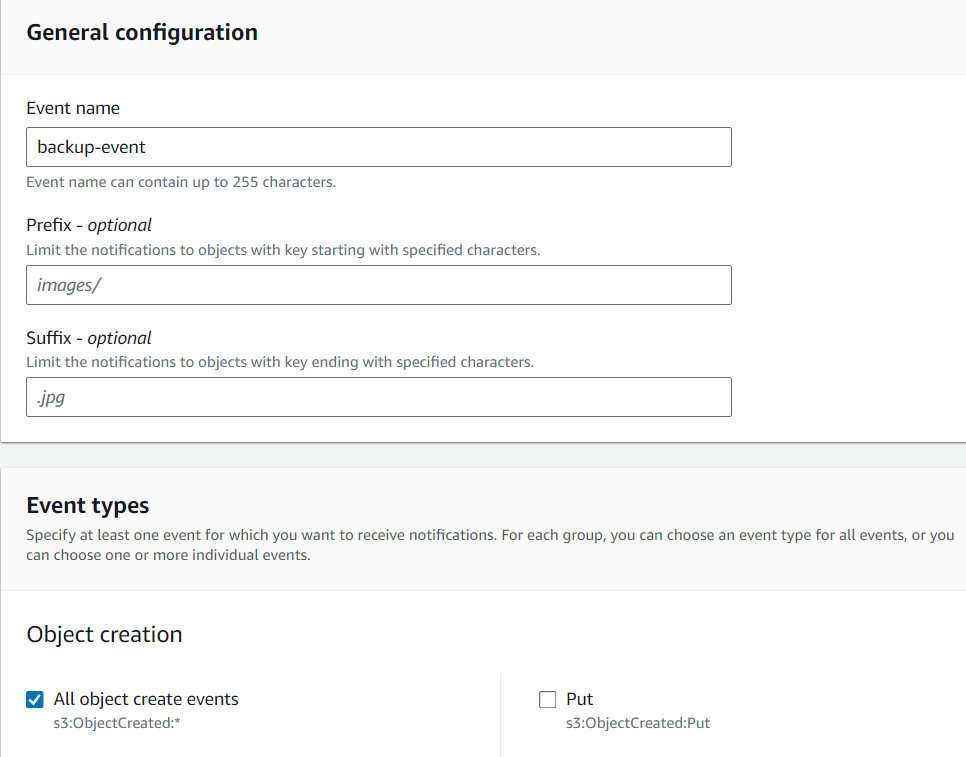
* + import json
  + import boto3
  + import urllib.parse
  + import time
  + s3\_client = boto3.client('s3')
  + def lambda\_handler(event, context):
  + print("Received event: " + json.dumps(event, indent=2))
  + # Get the bucket and object key from the event
  + try:
  + source\_bucket = event['Records'][0]['s3']['bucket']['name']
  + source\_key = event['Records'][0]['s3']['object']['key']
  + print(f"Original Source Key: {source\_key}")
  + source\_key = urllib.parse.unquote\_plus(source\_key)
  + print(f"Decoded Source Key: {source\_key}")
  + print(f"Source Bucket: {source\_bucket}")
  + except Exception as e:
  + print(f"Error parsing event: {e}")
  + raise e
  + # Define the backup bucket
  + backup\_bucket = 'my-backup-bucket-654'
  + # Adding a short delay to ensure the object is fully available
  + time.sleep(2)
  + # Verify the object exists in the source bucket
  + try:
  + s3\_client.head\_object(Bucket=source\_bucket, Key=source\_key)
  + print(f"Object {source\_key} exists in bucket {source\_bucket}")
  + except s3\_client.exceptions.NoSuchKey as e:
  + print(f"Object {source\_key} does not exist in bucket {source\_bucket}")
  + return {
  + 'statusCode': 404,
  + 'body': json.dumps(f"Object {source\_key} does not exist in bucket {source\_bucket}")
  + }
  + except Exception as e:
  + print(f"Error checking object: {e}")
  + raise e
  + # Copy the object
  + copy\_source = {'Bucket': source\_bucket, 'Key': source\_key}
  + try:
  + s3\_client.copy\_object(CopySource=copy\_source, Bucket=backup\_bucket, Key=source\_key)
  + print(f"Successfully copied {source\_key} from {source\_bucket} to {backup\_bucket}")
  + except Exception as e:
  + print(f"Error copying object: {e}")
  + raise e
  + return {
  + 'statusCode': 200,
  + 'body': json.dumps('Backup completed successfully!')
  + }
  + Click "Deploy" to save your changes.

**Step 4: Configure S3 Event Notification**

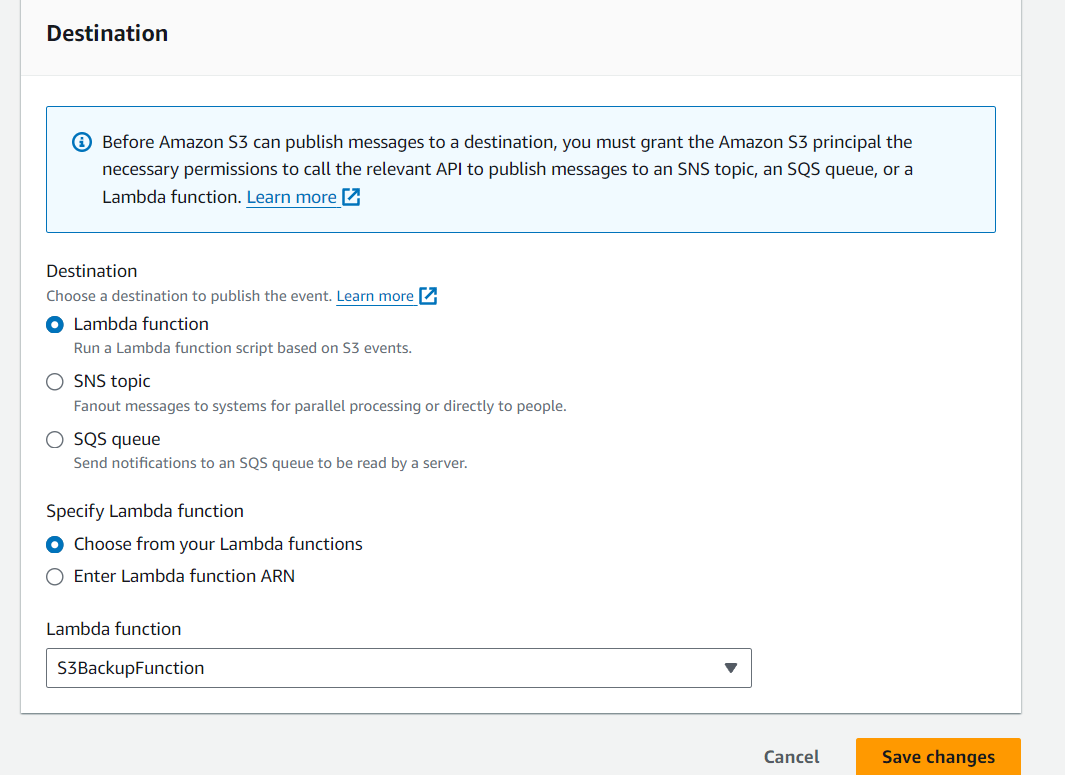
1. **Go to the S3 service**:
   * Navigate to the my-source-bucket-654.
2. **Set Up Event Notification**:
   * Click on the my-source-bucket-654.
   * Go to the "Properties" tab.



* + Scroll down to "Event notifications" and click "Create event notification".
  + Enter a name for the event, e.g., backup-event.



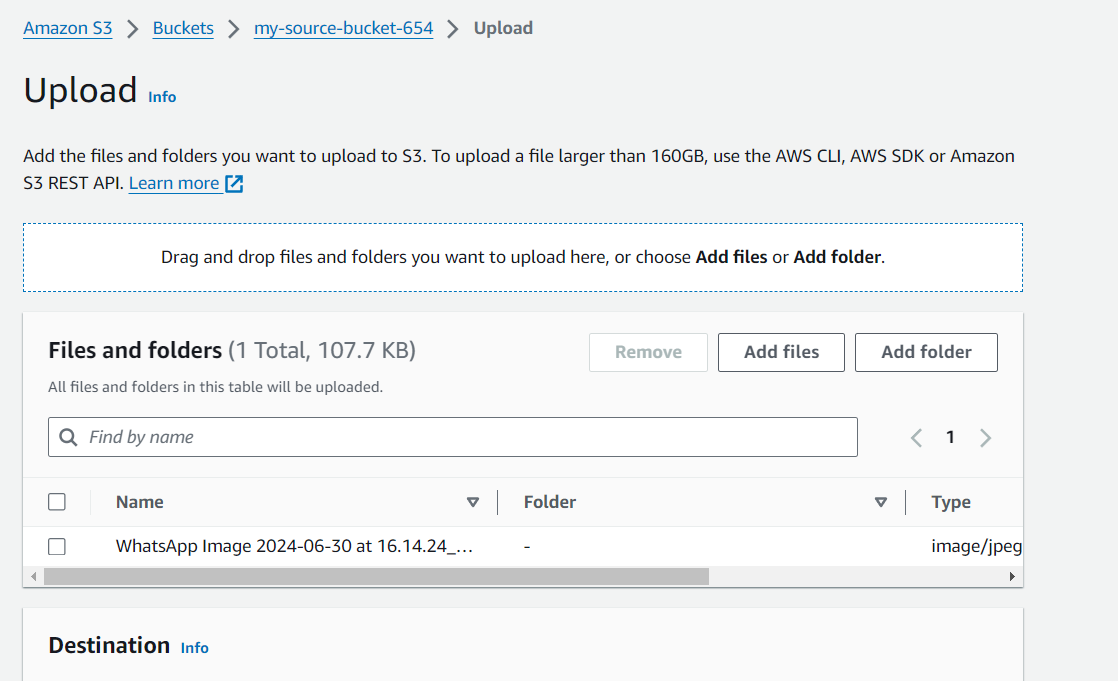
* + Under "Event types", select "All object create events".
  + Under "Destination", select "Lambda function".



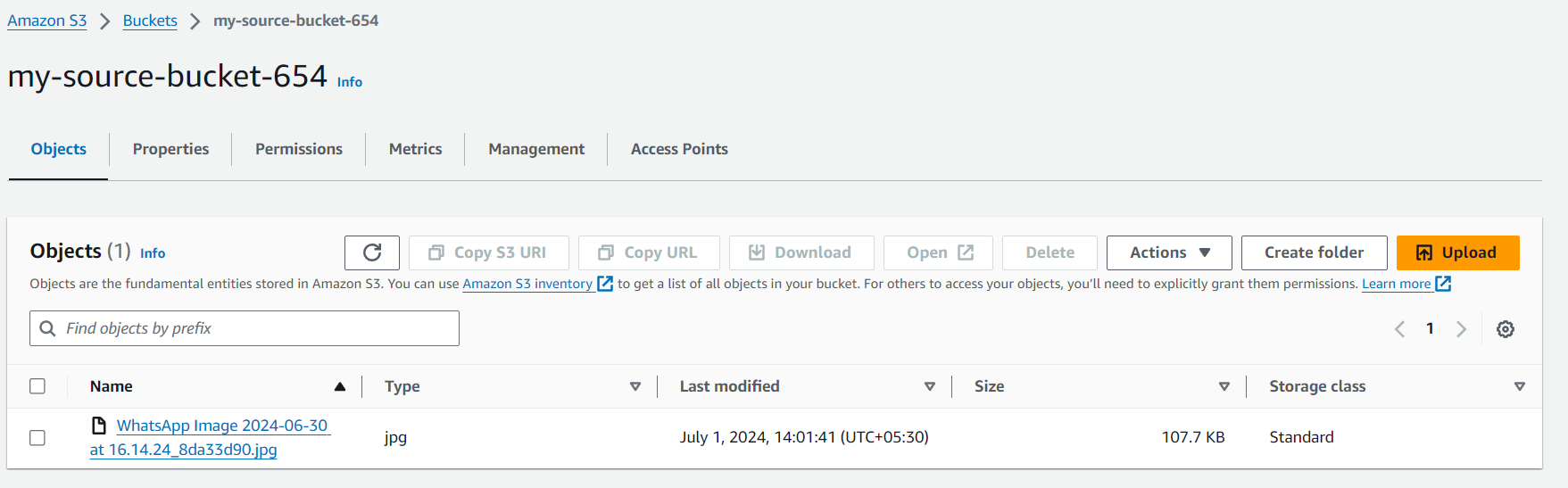
* + Choose the Lambda function created in Step 3 (S3BackupFunction).
  + Click "Save changes".

**Step 5: Deploy and Test**

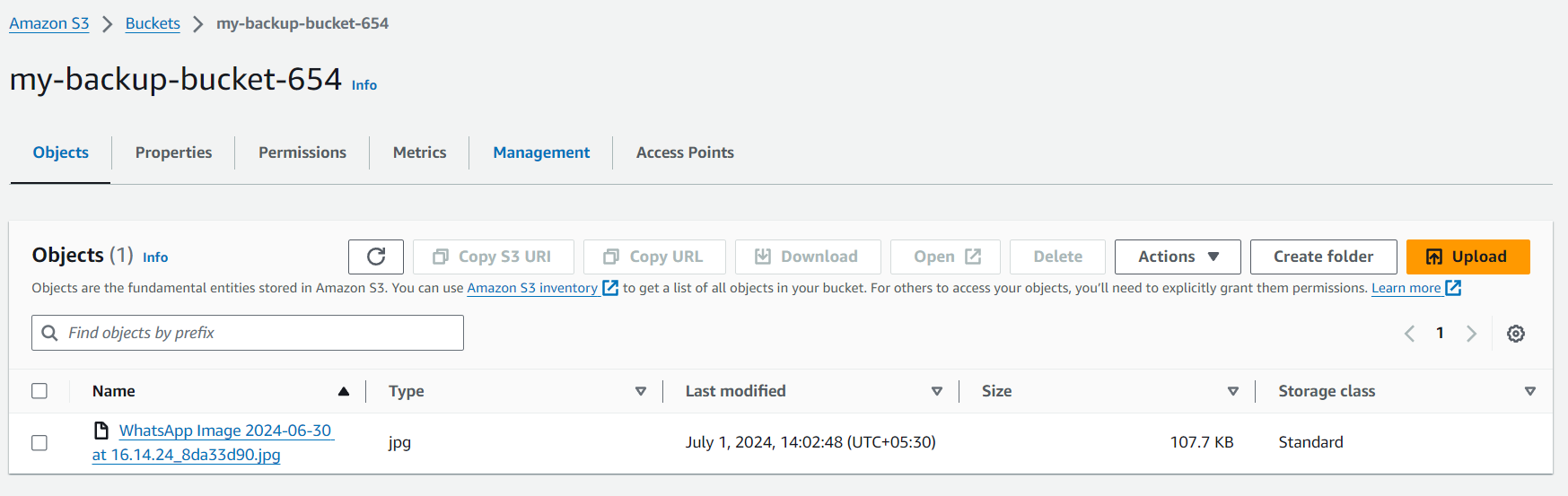
1. **Upload an Object to the Source Bucket**:
   * Go to the my-source-bucket-654.



* + Click "Upload" and add any file.



1. **Verify the Backup**:
   * Navigate to the my-backup-bucket-654.



* + Check if the uploaded file from the source-data-bucket is present in the my-backup-bucket-654.