Replicate data from primary bucket to backup bucket, when new file is uploaded in primary bucket, it should automatically reflect in backup bucket. Data should not be deleted from backup bucket even its deleted from primary bucket.

Steps:

Step1: Create Two S3 Buckets: **primary-bucket** and **backup-bucket**.

Step2: Create an IAM Role for the Lambda Function

- Go to the IAM service in the AWS Management Console.
- Click on "Roles" in the left-hand menu, then "Create role".
- Select "AWS service" and choose "Lambda".
- Click "Next: Permissions".
- Click "Create policy" and go to the JSON tab.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "s3:GetObject",
    "s3:ListBucket"
   ],
   "Resource": [
    "arn:aws:s3:::primary-bucket-7",
    "arn:aws:s3:::primary-bucket-7/*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "s3:PutObject"
   ],
   "Resource": [
    "arn:aws:s3:::backup-bucket-7/*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
```

```
            "sns:Publish"
          ],
          "Resource": "arn:aws:sns:ap-south-1:965519929135:s3DataBackUpSNS"
        }
      ]
    }
```

Attack the policy to Lambda role.

Step3: Create an SNS Topic and Subscribe Your Email.
Step4: Create a Lambda function.

```
import boto3
import urllib.parse

s3 = boto3.client('s3')
sns = boto3.client('sns')

def lambda_handler(event, context):
  for record in event['Records']:
    source_bucket = record['s3']['bucket']['name']
    source_key = urllib.parse.unquote_plus(record['s3']['object']['key'])
    destination_bucket = 'backup-bucket-7'
    copy_source = {'Bucket': source_bucket, 'Key': source_key}

    try:
      # Copy the object to the backup bucket
      s3.copy_object(CopySource=copy_source, Bucket=destination_bucket,
Key=source_key)
      print(f'Successfully copied {source_key} from {source_bucket} to
{destination_bucket}')

      # Send SNS notification
      sns.publish(
        TopicArn='arn:aws:sns:ap-south-1:965519929135:s3DataBackUpSNS',
        Subject='File Uploaded to Backup Bucket',
        Message=f'The file {source_key} has been successfully uploaded to
{destination_bucket}.'
      )
      print(f'Successfully sent SNS notification for {source_key}')
    except Exception as e:
      print(f'Error copying {source_key} from {source_bucket} to {destination_bucket}:
{str(e)}')
      raise e
```

The source bucket in the Lambda function is dynamically determined based on the event that triggers the function. This means that we don't need to hard-code the source bucket name in the Lambda function code. Instead, the source bucket is extracted from the event record whenever a new object is uploaded to the primary-bucket.

Step5: Configure S3 Event Notifications for the Primary Bucket

- Go to the S3 service in the AWS Management Console.
- Select the primary-bucket.
- Go to the "Properties" tab.
- Scroll down to "Event notifications" and click "Create event notification".
- Configure the event to trigger on s3:ObjectCreated:* and select the Lambda function S3ReplicationFunction.
- Save the event notification.

Step6: Configure S3 Event Notifications for the Backup Bucket

- Go to the S3 service in the AWS Management Console.
- Select the backup-bucket.
- Go to the "Properties" tab.
- Scroll down to "Event notifications" and click "Create event notification".
- Configure the event to trigger on s3:ObjectCreated:* and select the same Lambda function S3ReplicationFunction.
- Save the event notification.