Assignment 3 – S3

Updated demo recording: RDS_EC2_S3.mp4

ACCEPTANCE CRITERIA – Include the following in the PDF:

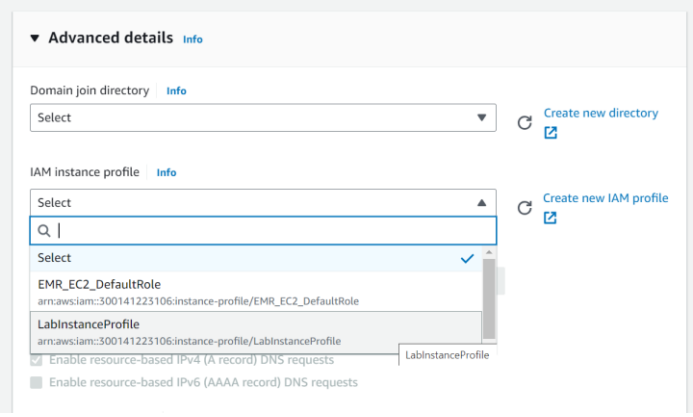
- Upload/download success in the EC2 terminal.
- SNS topic policy, the new event notification in the S3 bucket, and the email.
- Presigned URL generated by the lambda.

The goal of assignment 3: Practice some S3 features.

1. The first task is connecting an EC2 server with an S3 bucket that allows upload and download between S3 and EC2. Once you have that setup, any app on the EC2 can access S3. Another main thing in this task is to understand the usage of IAM. You always have to write IAM policies to integrate two distinct services in AWS. In the AWS Academy account, you just associate all services with the LabRole that already has preconfigured policies in place. If you do it in a regular AWS account, you must define the policy.
2. Task 2 offers insight into the workings of modern event-driven applications. What it does is, when you upload an object, that triggers a new-object-created event and sends that event to SNS (you used SNS in lab 2). Then SNS captures the event and sends you an email. Notice, that it is all event-driven.
3. A signed URL is the most commonly used S3 feature in practice. It facilitates the temporary downloading and uploading of files for a specified duration, addressing the requirements of the majority of real-world applications.

Task 1 – Download/Upload a file from S3 in EC2

a. Create an **S3 bucket** and put a file in it.
b. Go to the EC2 console. Create an EC2 instance. While launching, expand the "Advanced details" and select the LabInstanceProfile (it is the LabRole). Or In a regular AWS account, create an IAM role for the EC2 with S3 permissions.



c. SSH into it like you did in assignment 1. Download and upload the file AWS CLI. Following the CLI command download the file in S3 into the EC2 instance. The CP command's first argument is the source and the second argument is the destination. If you swap values, it will upload. Depending on the AWS CLI version of the EC2 instance, this command may need to be updated.

```
aws s3 cp s3://<bucket_name>/<file_name_in_s3> <new_file_name_in_s3>
```

Task 2 – S3 Event Notification

Send an email to yourself when the object is created in the bucket with the S3 event notification feature and SNS. You should already be familiar with the SNS that you used to send an email notification in the previous lab when setting the billing alarm.

1. You need to create an SNS topic with a subscriber (your email).
2. You **must** write a **resource-based policy** in the SNS topic **after creation**. Remember, if you are connecting 2 different AWS services, you must write an IAM policy. In this case, you have to write a resource-based policy on the SNS topic that allows the S3 to publish messages on the topic. **[Include the IAM policy in the PDF].** You **must replace** the
    - Resource with the SNS topic ARN
    - AWS:SourceAccount with the AWS account you see in the console (part of ARNs)
    - Aws:SourceArn with the bucket ARN

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": "SNS:Publish",
      "Resource": "arn:aws:sns:us-east-1:475249589989:MyS3Topic",
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "475249589989"
        },
        "ArnLike": {
          "AWS:SourceArn": "arn:aws:s3:::csnov516demo"
        }
      }
    }
  ]
}
```

3. Create an S3 bucket or use an existing one. Go to the Properties tab. In the “Event notifications” section, click on the “Create event notifications”.
    a. Give event name
    b. Select All object create events
    c. In Destination, select the SNS topic then choose the SNS you created earlier. If the policy in the SNS is correct, you should be able to create it. If you get an error, fix that and it will work!

Task 3 – S3 Signed URL

Write a lambda that returns a Signed URL that downloads an object in S3. Create a "Test Event" to trigger the lambda and test your code. For more: https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/index.html Add the following to your lambda and run your test (replace the bucket and key). The key is your file name in S3. **Don't copy-paste this. Write it yourself.**

```
import { S3Client } from '@aws-sdk/client-s3';
import { GetObjectCommand } from '@aws-sdk/client-s3';
import { getSignedUrl } from '@aws-sdk/s3-request-presigner';
const s3 = new S3Client({ region: 'us-east-1' });

export const handler = async (event) => {
    const params = { Bucket: '<bucket_name>', Key: '<object_key>' };
    const command = new GetObjectCommand(params);
    const url = await getSignedUrl(s3, command, { expiresIn: 3600 });
    return url;
  }
```