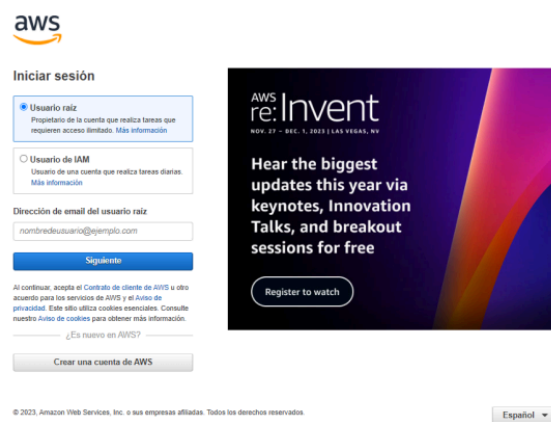


Guía para la solución del subreto 2.

Iniciar sesión en la consola de AWS.

[consola AWS](#)

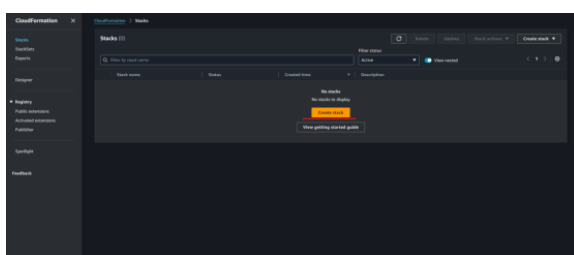
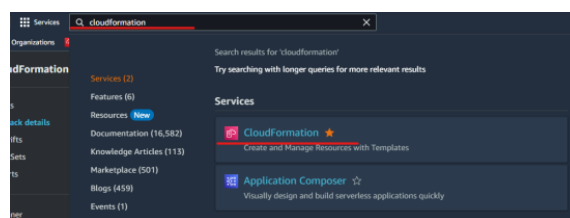


```
1 Resources:
2   # Bucket de S3
3   LambdaCodeBucket:
4     Type: AWS::S3::Bucket
5     Properties:
6       BucketName: compumundo-lambda-code
```

Crear archivo con extensión **.yml** en cualquier ubicación de su pc y agregar la siguiente configuración que permite la creación de un **bucket de S3** para almacenar el código en **Nodejs** que se usará para agregarlo a las lambdas.

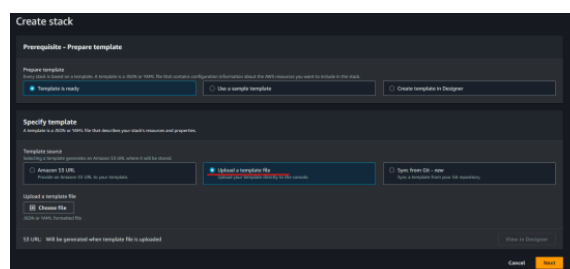
Nota: Se debe asegurar que el **bucketName** sea un valor **único a nivel global** en AWS.

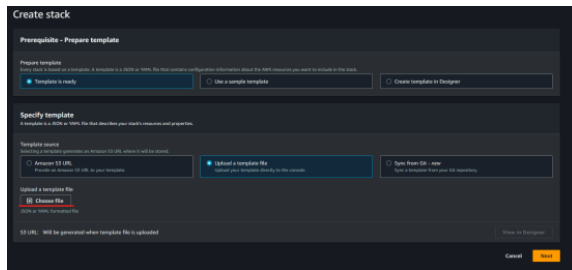
En la consola de AWS seleccionar el archivo de Cloudformation.



Se debe hacer clic en **Create stack**.

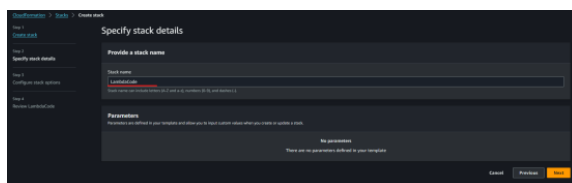
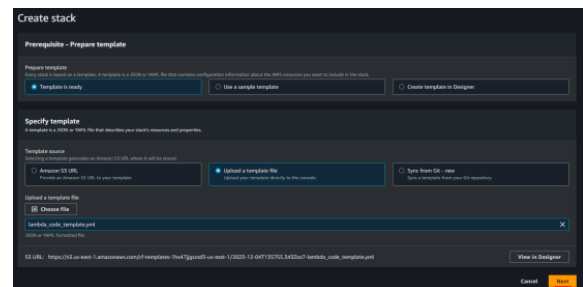
Seleccionar la opción **Upload a template file**.





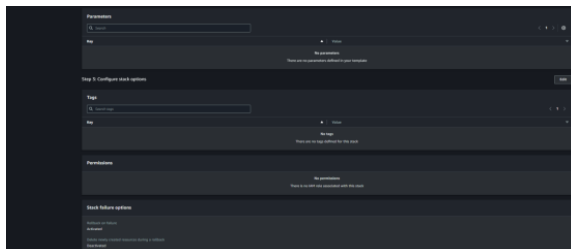
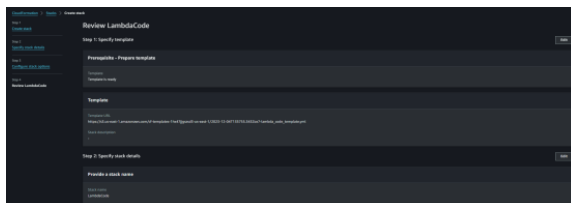
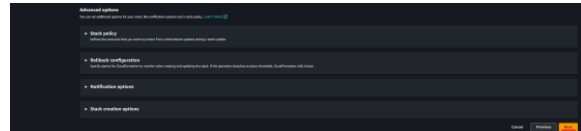
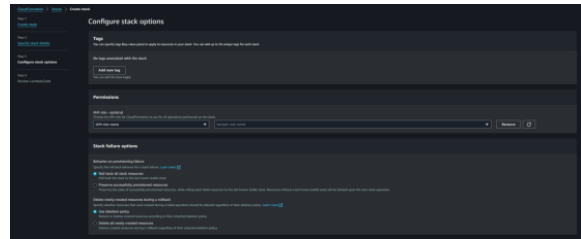
Hacer clic en el botón **Choose file** y seleccionar el archivo con extensión **.yml** que se creó en el paso número 2.

Hacemos clic en **Next**.

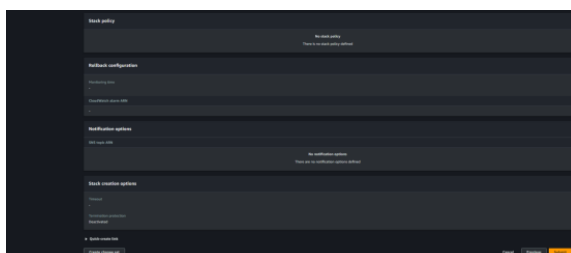


Asignamos un nombre al stack y hacemos clic en **Next**.

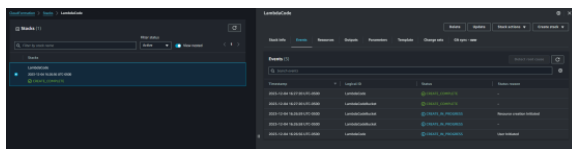
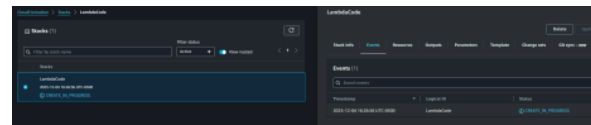
Las siguientes opciones las dejamos como están ya preestablecidas y hacemos clic en **Next**.



Nos mostrará un resumen del stack, validamos que todo esté correcto y hacemos clic en **Submit**.

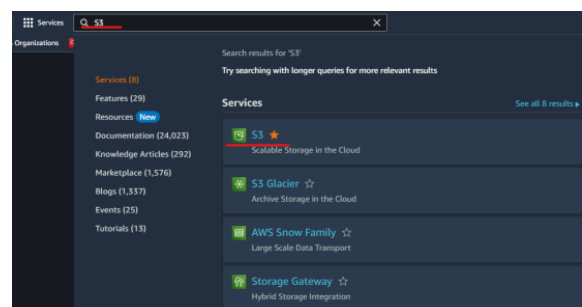


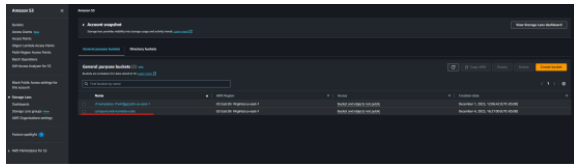
Se inicia el proceso de ejecución del stack para la creación del bucket de S3 en donde se va a almacenar el código de las lambdas.



Una vez que finalice la ejecución debe aparecer el status **CREATE_COMPLETE** y ya podemos ir al servicio de S3 a validar el bucket creado.

Seleccionar el servicio de S3.

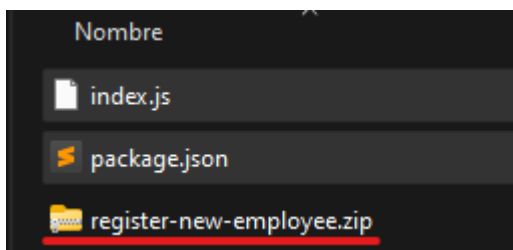




Se puede evidenciar que se ha creado el bucket de S3.

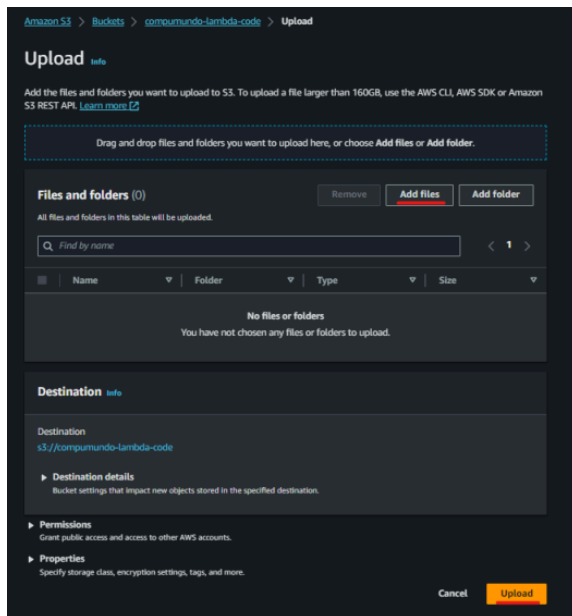
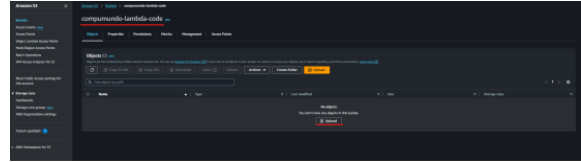
En cualquier ubicación de su pc debe crear un nuevo proyecto de **Nodejs** para escribir el siguiente código de la lambda que permitirá recibir un json de entrada con id (cédula del empleado) y un archivo en base64. El archivo debe guardarse en el bucket **active-employee-files**, asignando al nombre del archivo la cédula.

```
1  const { S3 } = require("aws-sdk");
2
3  exports.handler = async (event, context) => {
4    const body = JSON.parse(event.body);
5    const id = body.id;
6    const base64File = body.file;
7    const fileBuffer = Buffer.from(base64File, "base64");
8    const s3 = new S3();
9    const params = {
10     Bucket: "active-employee-files",
11     Key: id,
12     Body: fileBuffer,
13     ContentType: "application/pdf",
14   };
15   await s3.putObject(params).promise();
16   return {
17     statusCode: 200,
18     body: JSON.stringify({
19       message: "El archivo se cargó correctamente",
20     }),
21   };
22 }
```



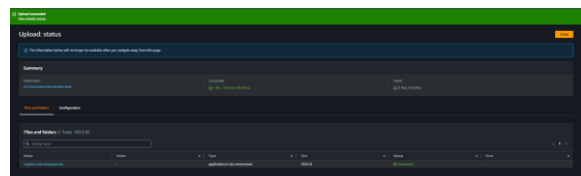
Se debe comprimir el index.js y el package.json en un archivo con extensión .zip.

Debe subir el archivo .zip al bucket de S3 previamente creado (**compumundo-lambda-code**) haciendo clic en **Upload**.



Hacer clic en **Add files** y clic en **Upload**.

Una vez cargado debe salir el mensaje **“Upload succeeded”**.



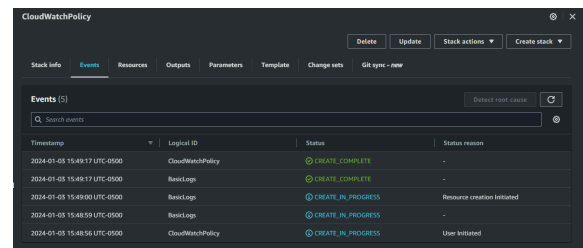
```

1  Resources:
2    CloudWatchPolicy:
3      Type: AWS::IAM::ManagedPolicy
4      Properties:
5        PolicyDocument:
6          Version: 2012-10-17
7          Statement:
8            - Effect: Allow
9              Action:
10               - "logs:CreateLogGroup"
11               - "logs:CreateLogStream"
12               - "logs:PutLogEvents"
13            Resource: "*"

```

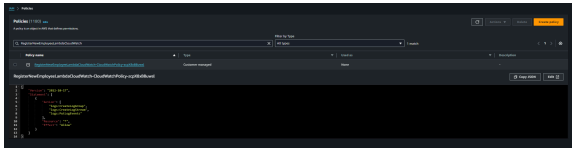
Creamos un nuevo archivo con extensión .yaml que nos permitirá construir la política para que permita dejar logs en CloudWatch.

Ingresamos a CloudFormation y cargamos el archivo .yaml y ejecutamos el stack.



The screenshot shows the AWS CloudFormation console interface. At the top, there are tabs for 'Stack info', 'Events', 'Resources', 'Outputs', 'Parameters', 'Template', 'Change sets', and 'Git sync - new'. The 'Events' tab is selected, displaying a table of events for the 'CloudWatchPolicy' stack. The table has columns for 'Timestamp', 'Logical ID', 'Status', and 'Status reason'. The events listed are:

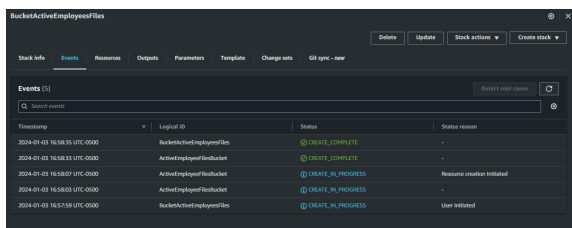
| Timestamp | Logical ID | Status | Status reason |
|------------------------------|------------------|--------------------|-----------------------------|
| 2024-01-03 15:49:17 UTC-0500 | CloudWatchPolicy | CREATE_COMPLETE | - |
| 2024-01-03 15:49:17 UTC-0500 | BaseLog | CREATE_COMPLETE | - |
| 2024-01-03 15:49:00 UTC-0500 | BaseLog | CREATE_IN_PROGRESS | Resource creation initiated |
| 2024-01-03 15:48:59 UTC-0500 | BaseLog | CREATE_IN_PROGRESS | - |
| 2024-01-03 15:48:56 UTC-0500 | CloudWatchPolicy | CREATE_IN_PROGRESS | User initiated |



Validamos que la política se haya creado correctamente.

Creamos un nuevo archivo con extensión .yaml que nos permitirá adicionar un bucket de S3 con el nombre active-employee-files, en el cual, se almacenará el archivo que ingresamos en la lambda register-new-employee. El nombre del archivo debe ser el id que se envía en el evento de la lambda de registro.

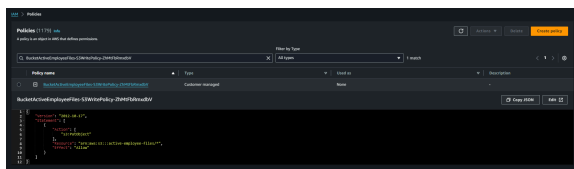
```
1 Resources:
2   # Bucket de S3
3   ActiveEmployeeFilesBucket:
4     Type: AWS::S3::Bucket
5     Properties:
6       BucketName: active-employee-files
```



Ingresamos a CloudFormation y cargamos el archivo .yaml y ejecutamos el stack.

Creamos un nuevo archivo con extensión .yml que nos permitirá agregar la política para guardar archivos en el bucket de S3 active-employee-files.

```
1 Resources:
2   S3WritePolicy:
3     Type: AWS::IAM::ManagedPolicy
4     Properties:
5       PolicyDocument:
6         Version: 2012-10-17
7         Statement:
8           - Effect: Allow
9             Action:
10              - s3:PutObject
11              Resource: arn:aws:s3:::active-employee-files/*
```

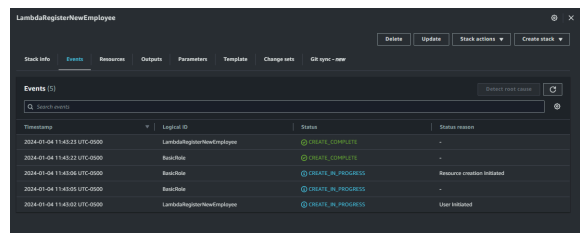


Validamos que se haya creado la política correctamente.

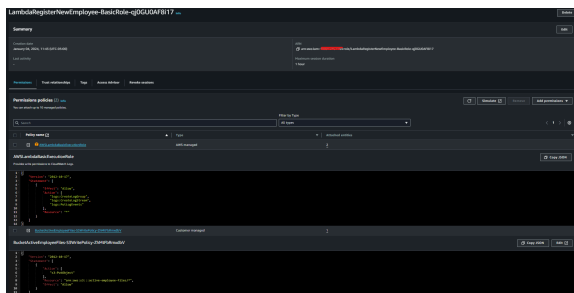
Creamos la plantilla .yml para el role de la lambda register-new-employee, agregando la política para almacenar archivos en el bucket de S3 active-employee-files y escribir logs en CloudWatch.

```
1 Resources:
2   BasicRole:
3     Type: AWS::IAM::Role
4     Properties:
5       AssumeRolePolicyDocument:
6         Version: 2012-10-17
7         Statement:
8           - Effect: Allow
9             Principal:
10               Service: lambda.amazonaws.com
11             Action: sts:AssumeRole
12       ManagedPolicyArns:
13         - arn:aws:iam::[redacted]:policy/RegisterNewEmployeeLambdaCloudWatch-CloudWatchPolicy-zcpX9x8Suwsl
14         - arn:aws:iam::[redacted]:policy/BucketActiveEmployeeFiles-S3WritePolicy-zhMFPBxwdbV
```

Cargamos la plantilla en Cloudformation.



| Timestamp | Logical ID | Status | Status reason |
|------------------------------|---------------------------|--------------------|-----------------------------|
| 2024-01-04 11:40:22 UTC-0500 | LambdaRegisterNewEmployee | CREATE_COMPLETE | - |
| 2024-01-04 11:40:22 UTC-0500 | BasicRole | CREATE_COMPLETE | - |
| 2024-01-04 11:40:26 UTC-0500 | BasicRole | CREATE_IN_PROGRESS | Resource creation initiated |
| 2024-01-04 11:40:26 UTC-0500 | BasicRole | CREATE_IN_PROGRESS | - |
| 2024-01-04 11:40:26 UTC-0500 | LambdaRegisterNewEmployee | CREATE_IN_PROGRESS | User initiated |



Validamos que se haya creado correctamente el role y que se tenga asociadas las dos políticas (Cloudwatch y S3).

```

1 Resources:
2   # Función de Lambda
3   RegisterNewEmployeeLambda:
4     Type: AWS::Lambda::Function
5     Properties:
6       FunctionName: register-new-employee
7       Role: arn:aws:iam::XXXXXXXXXX:role/CloudWatchPolicy-BasicLogs-U1TdDEkFBTEJ
8       Handler: index.handler
9       Runtime: nodejs16.x
10      Code:
11        S3Bucket: compumundo-lambda-code
12        S3Key: register-new-employee.zip
13      Timeout: 29

```

Creamos un nuevo archivo con extensión .yml que nos permitirá construir la lambda para el registro de nuevos empleados, asignando el rol que creamos previamente y asociamos el bucket y el archivo comprimido .zip en donde se encuentra nuestro código.

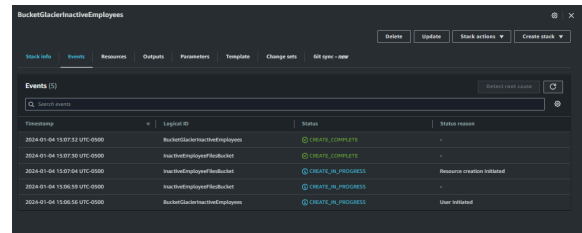
```

1 Resources:
2   # Bucket de Glacier
3   InactiveEmployeeFilesBucket:
4     Type: AWS::S3::Bucket
5     Properties:
6       BucketName: inactive-employee-files
7       LifecycleConfiguration:
8         Rules:
9           - Id: ExpirationRule
10             Prefix: ""
11             Status: Enabled
12             ExpirationInDays: 3650

```

Creamos la plantilla .yml para la creación del bucket tipo Glacier inactive-employee-files.

Cargamos la plantilla en cloudformation y la ejecutamos.



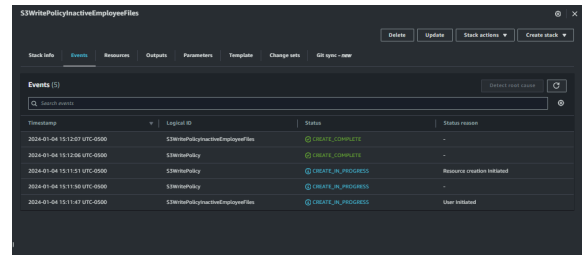
The screenshot shows the AWS CloudFormation console for a stack named 'BucketGlacierInactiveEmployees'. The 'Events' tab is selected, displaying a list of events. The events show the successful creation of the stack and its resources, including the 'BucketGlacierInactiveEmployees' stack itself, the 'InactiveEmployeeFilesBucket' bucket, and the 'ActiveEmployeeFilesBucket' bucket. The status of all events is 'CREATE_COMPLETE'.

| Timestamp | Logical ID | Status | Status reason |
|------------------------------|--------------------------------|--------------------|-----------------------------|
| 2024-01-04 15:07:15 UTC-0500 | BucketGlacierInactiveEmployees | CREATE_COMPLETE | - |
| 2024-01-04 15:07:15 UTC-0500 | InactiveEmployeeFilesBucket | CREATE_COMPLETE | - |
| 2024-01-04 15:07:15 UTC-0500 | ActiveEmployeeFilesBucket | CREATE_IN_PROGRESS | Resource creation initiated |
| 2024-01-04 15:07:15 UTC-0500 | ActiveEmployeeFilesBucket | CREATE_IN_PROGRESS | - |
| 2024-01-04 15:07:15 UTC-0500 | ActiveEmployeeFilesBucket | CREATE_IN_PROGRESS | - |

Creamos la plantilla .yml para la creación de la política para obtener y eliminar archivos del bucket de S3 active-employee-files y para guardar archivos en el bucket S3 inactive-employee-files.

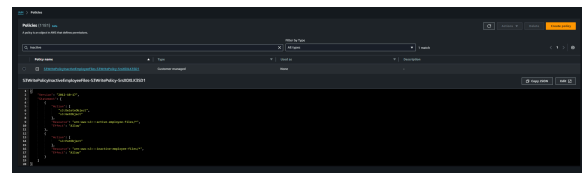
```
1 Resources:
2   S3WritePolicy:
3     Type: AWS::IAM::ManagedPolicy
4     Properties:
5       PolicyDocument:
6         Version: 2012-10-17
7         Statement:
8           - Effect: Allow
9             Action:
10              - s3:DeleteObject
11              - s3:GetObject
12             Resource: arn:aws:s3:::active-employee-files/*
13           - Effect: Allow
14             Action:
15              - s3:PutObject
16             Resource: arn:aws:s3:::inactive-employee-files/*
```

Cargamos la plantilla en cloudformation y la ejecutamos.



| Timestamp | Logical ID | Status | Status reason |
|------------------------------|------------------------------------|--------------------|-----------------------------|
| 2024-01-04 15:12:07 UTC-0500 | S3WritePolicyInactiveEmployeeFiles | CREATE_COMPLETE | - |
| 2024-01-04 15:12:06 UTC-0500 | S3WritePolicy | CREATE_COMPLETE | - |
| 2024-01-04 15:12:01 UTC-0500 | S3WritePolicy | CREATE_IN_PROGRESS | - |
| 2024-01-04 15:11:50 UTC-0500 | S3WritePolicy | CREATE_IN_PROGRESS | Resource creation initiated |
| 2024-01-04 15:11:47 UTC-0500 | S3WritePolicyInactiveEmployeeFiles | CREATE_IN_PROGRESS | User initiated |

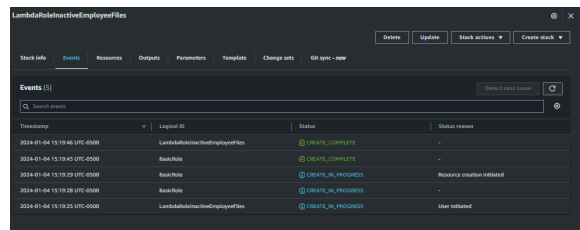
Validamos que se haya creado correctamente la política.



Creamos plantilla .yaml para la creación del role al que le vamos a asignar las políticas que hemos creado anteriormente.

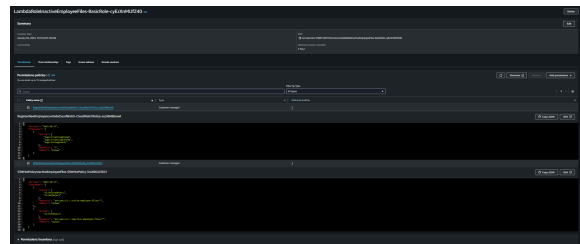
```
1 Resources:
2   BasicRole:
3     Type: AWS::IAM::Role
4     Properties:
5       AssumeRolePolicyDocument:
6         Version: 2012-10-17
7         Statement:
8           - Effect: Allow
9             Principal:
10              Service: lambda.amazonaws.com
11            Action: sts:AssumeRole
12       ManagedPolicyArns:
13         - arn:aws:iam::[redacted]:policy/RegisterNewEmployeeLambdaCloudWatch-CloudWatchPolicy-zcpX8xBBuwl
14         - arn:aws:iam::[redacted]:policy/S3WritePolicyInactiveEmployeeFiles-S3WritePolicy-SrsX8XKJ3D1
```

Cargamos la plantilla en cloudformation y la ejecutamos.



| Timestamp | Logical ID | Status | Status reason |
|------------------------------|---------------------------------|--------------------|-----------------------------|
| 2024-01-04 13:19:46 UTC-0500 | LambdaRoleInactiveEmployeeFiles | CREATE_COMPLETE | - |
| 2024-01-04 13:19:45 UTC-0500 | BaseRole | CREATE_COMPLETE | - |
| 2024-01-04 13:19:23 UTC-0500 | BaseRole | CREATE_IN_PROGRESS | Resource creation initiated |
| 2024-01-04 13:19:23 UTC-0500 | BaseRole | CREATE_IN_PROGRESS | - |
| 2024-01-04 13:19:23 UTC-0500 | LambdaRoleInactiveEmployeeFiles | CREATE_IN_PROGRESS | User initiated |

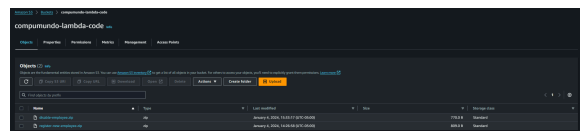
Validamos que se haya creado correctamente el role.



Escribimos el código de la lambda para inactivar empleados.

```
1  const AWS = require("aws-sdk");
2  const s3 = new AWS.S3();
3
4  exports.handler = async (event, context) => {
5    const body = JSON.parse(event.body);
6    const id = body.id;
7    const object = await s3.getObject({
8      Bucket: "active-employee-files",
9      Key: id,
10    });
11    const params = {
12      Bucket: "inactive-employee-files",
13      Key: id,
14      Body: object.Body,
15      ContentType: "application/pdf",
16    };
17    await s3.putObject(params).promise();
18    await s3.deleteObject({
19      Bucket: "active-employee-files",
20      Key: id,
21    }).promise();
22    return {
23      statusCode: 200,
24      body: JSON.stringify({
25        message: "El empleado ha sido inactivado exitosamente.",
26      })
27    };
28  };
29
```

Comprimimos en un archivo .zip el index.js y el package.json y lo almacenamos en el bucket compumundo-lambda-code.



Creamos la plantilla .yml para la creación de la lambda disable-employee, asociando el role que se creó previamente y el archivo comprimido en formato .zip donde se encuentra el código fuente.

```
1  Resources:
2    # Función de Lambda
3    DisableEmployeeLambda:
4      Type: AWS::Lambda::Function
5      Properties:
6        FunctionName: disable-employee
7        Role: arn:aws:iam::226654301204:role/LambdaRoleInactiveEmployeeFiles-BasicRole-cyExnMUfZu0
8        Handler: index.handler
9        Runtime: nodejs16.x
10       Code:
11         S3Bucket: compumundo-lambda-code
12         S3Key: disable-employee.zip
13       Timeout: 29
```


| Timestamp | Logical ID | Status | Status message |
|------------------------------|------------------------------|--------------------|-----------------------------|
| 2024-01-04 15:39:33 UTC-0500 | LambdaFunctionStatusEmployee | CREATE_COMPLETE | - |
| 2024-01-04 15:39:32 UTC-0500 | EmployeeRestoreStatusTable | CREATE_COMPLETE | - |
| 2024-01-04 15:39:41 UTC-0500 | EmployeeRestoreStatusTable | CREATE_IN_PROGRESS | Resource creation initiated |
| 2024-01-04 15:39:41 UTC-0500 | EmployeeRestoreStatusTable | CREATE_IN_PROGRESS | - |
| 2024-01-04 15:39:42 UTC-0500 | LambdaFunctionStatusEmployee | CREATE_IN_PROGRESS | User initiated |

Cargamos en cloudformation la plantilla y la ejecutamos.

Creamos una nueva plantilla .yaml para la creación de una tabla de DynamoDB llamada employee-restore-status con llave de partición id (string), un índice secundario global (GSI) con llave de partición status (string) y con TTL enabled.

```

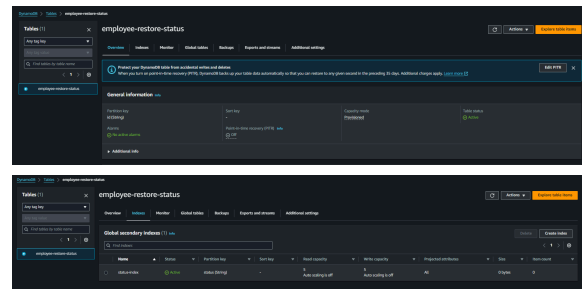
1  Resources:
2      EmployeeRestoreStatusTable:
3          Type: AWS::DynamoDB::Table
4          Properties:
5              TableName: employee-restore-status
6              AttributeDefinitions:
7                  - AttributeName: id
8                    AttributeType: S
9                  - AttributeName: status
10                     AttributeType: S
11              KeySchema:
12                  - AttributeName: id
13                    KeyType: HASH
14              ProvisionedThroughput:
15                  ReadCapacityUnits: 5
16                  WriteCapacityUnits: 5
17              GlobalSecondaryIndexes:
18                  - IndexName: status-index
19                    KeySchema:
20                        - AttributeName: status
21                          KeyType: HASH
22                  Projection:
23                      ProjectionType: ALL
24              ProvisionedThroughput:
25                  ReadCapacityUnits: 5
26                  WriteCapacityUnits: 5
27              TimeToLiveSpecification:
28                  Enabled: true
29                  AttributeName: ttl

```

| Timestamp | Logical ID | Status | Status message |
|------------------------------|----------------------------|--------------------|-----------------------------|
| 2024-01-04 16:09:45 UTC-0500 | DynamoDBRestoreStatus | CREATE_COMPLETE | - |
| 2024-01-04 16:09:44 UTC-0500 | EmployeeRestoreStatusTable | CREATE_COMPLETE | - |
| 2024-01-04 16:09:22 UTC-0500 | EmployeeRestoreStatusTable | CREATE_IN_PROGRESS | Resource creation initiated |
| 2024-01-04 16:09:21 UTC-0500 | EmployeeRestoreStatusTable | CREATE_IN_PROGRESS | - |
| 2024-01-04 16:09:18 UTC-0500 | DynamoDBRestoreStatus | CREATE_IN_PROGRESS | User initiated |

Cargamos en cloudformation la plantilla y la ejecutamos.

Validamos que se haya creado correctamente la tabla en DynamoDB.



```
1 Resources:
2   DynamoDBPolicy:
3     Type: AWS::IAM::ManagedPolicy
4     Properties:
5       PolicyDocument:
6         Version: 2012-10-17
7         Statement:
8           - Effect: Allow
9             Action:
10              - dynamodb:PutItem
11              Resource: arn:aws:dynamodb:us-east-1:XXXXXXXXXX:table/employee-restore-status
```

```
1 Resources:
2   S3WritePolicy:
3     Type: AWS::IAM::ManagedPolicy
4     Properties:
5       PolicyDocument:
6         Version: 2012-10-17
7         Statement:
8           - Effect: Allow
9             Action:
10              - s3:PutObject
11              Resource: arn:aws:s3:::active-employee-files/*
12           - Effect: Allow
13             Action:
14              - s3:GetObject
15              - s3>DeleteObject
16              Resource: arn:aws:s3:::inactive-employee-files/*
```

Creamos la plantilla .yaml para la creación de la política que permite guardar items en la tabla de DynamoDB employee-restore-status y la política de S3 para poder recuperar los archivos.

Cargamos la plantilla en cloudformation y la ejecutamos.

DynamoDB

DeleteUpdateStack actions▼Create stack▼

Stack infoEventsResourcesOutputsParametersTemplateChange setsGit sync - new

Events (5)

Detect root cause

Search events

| Timestamp | Logical ID | Status | Status reason |
|------------------------------|----------------|--------------------|-----------------------------|
| 2024-01-12 09:00:29 UTC-0500 | DynamoDB | CREATE_COMPLETE | - |
| 2024-01-12 09:00:28 UTC-0500 | DynamoDBPolicy | CREATE_COMPLETE | - |
| 2024-01-12 09:00:13 UTC-0500 | DynamoDBPolicy | CREATE_IN_PROGRESS | Resource creation initiated |
| 2024-01-12 09:00:11 UTC-0500 | DynamoDBPolicy | CREATE_IN_PROGRESS | - |
| 2024-01-12 09:00:08 UTC-0500 | DynamoDB | CREATE_IN_PROGRESS | User initiated |

S3StartRecoverEmployee

DeleteUpdateStack actions▼Create stack▼

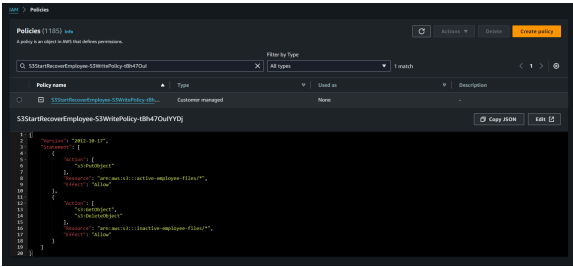
Stack infoEventsResourcesOutputsParametersTemplateChange setsGit sync - new

Events (5)

Detect root cause

Search events

| Timestamp | Logical ID | Status | Status reason |
|------------------------------|------------------------|--------------------|-----------------------------|
| 2024-01-15 11:59:04 UTC-0500 | S3StartRecoverEmployee | CREATE_COMPLETE | - |
| 2024-01-15 11:59:03 UTC-0500 | S3RecoveryPolicy | CREATE_COMPLETE | - |
| 2024-01-15 11:58:47 UTC-0500 | S3RecoveryPolicy | CREATE_IN_PROGRESS | Resource creation initiated |
| 2024-01-15 11:58:46 UTC-0500 | S3RecoveryPolicy | CREATE_IN_PROGRESS | - |
| 2024-01-15 11:58:44 UTC-0500 | S3StartRecoverEmployee | CREATE_IN_PROGRESS | User initiated |

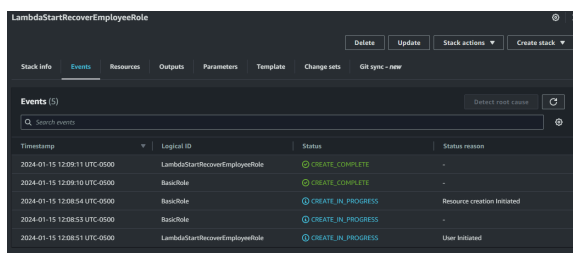


Creamos plantilla .yml para la creación del role al que le vamos a asignar las políticas que hemos creado anteriormente.

```

1 Resources:
2   BasicRole:
3     Type: AWS::IAM::Role
4   Properties:
5     AssumeRolePolicyDocument:
6       Version: 2012-10-17
7     Statement:
8       - Effect: Allow
9         Principal:
10            Service: lambda.amazonaws.com
11        Action: sts:AssumeRole
12
13 ManagedPolicies:
14   - arn:aws:iam::[redacted]:policy/RegisterNewEmployees-awscloudwatch-policy-zcp8xb8wsl
15   - arn:aws:iam::[redacted]:policy/DynatracePolicy-8-nsctcm2w
16   - arn:aws:iam::[redacted]:policy/S3StartRecoverEmployee-SMwritePolicy-tbhr0u1Yvj

```



Cargamos la plantilla en cloudformation y la ejecutamos.

Validamos que se haya creado correctamente el role.

The screenshot displays the AWS IAM console interface for the role **LambdaStartRecoverEmployeeRole-BasicRole-SHOWefnL8Y0**. The left sidebar shows the navigation menu with options like Dashboard, Access management, User groups, Users, Roles, Policies, Identity providers, Account settings, Access reports, Access analyzer, External access, Unused access, and Analyzer settings. The main content area is divided into several sections:

- Summary:** Shows the role name, creation date (January 15, 2024, 12:08 UTC-05:00), and the ARN: `arn:aws:iam::111111111111:role/LambdaStartRecoverEmployeeRole-BasicRole-SHOWefnL8Y0`. It also indicates the last activity and maximum session duration (1 hour).
- Permissions:** A section with tabs for Permissions, Trust relationships, Tags, Access Advisor, and Provider sessions. It includes a search bar and a filter by type dropdown.
- Permissions policies:** A list of attached policies, including **DynamoDB-DynamoDBPolicy-AccessMKZxW**, **RegisterNewEmployeeLambdaCloudWatch-CloudWatchPolicy-azpKbdlBwld**, and **SSStartRecoverEmployeeSSWitPolicy-lBh47OufYDj**. Each policy entry shows its name, version, and a "Copy JSON" button.
- Permissions boundary:** A section indicating that no boundary is currently set.
- Generate policy based on CloudTrail events:** A section with a "Generate policy" button and a note that no records were found for the past 7 days.

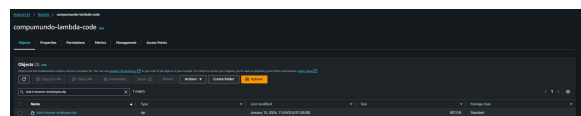
```

1  const AWS = require("aws-sdk");
2  const s3 = new AWS.S3();
3  const dynamodb = new AWS.DynamoDB();
4
5  exports.handler = async (event, context) => {
6    const body = JSON.parse(event.body);
7    const id = body.id;
8    const object = await s3.getObject({
9      Bucket: "inactive-employee-files",
10     Key: id,
11   });
12   const params = {
13     Bucket: "active-employee-files",
14     Key: id,
15     Body: object.Body,
16     ContentType: "application/pdf",
17   };
18   await s3.putObject(params).promise();
19   await s3.deleteObject({
20     Bucket: "inactive-employee-files",
21     Key: id,
22   }).promise();
23   const paramsItem = {
24     TableName: 'employee-restore-status',
25     Item: {
26       id: {
27         S: id
28       },
29       status: {
30         S: "IN PROGRESS"
31       },
32       ttl: {
33         S: "1800"
34       }
35     }
36   };
37   await dynamodb.putItem(paramsItem).promise();
38   return {
39     statusCode: 200,
40     body: JSON.stringify({
41       message: "El archivo se recuperó correctamente",
42     }),
43   };
44 };

```

Escribimos el código de la lambda start-recover-employee.

Comprimimos en un archivo .zip el index.js y el package.json y lo almacenamos en el bucket compumundo-lambda-code.



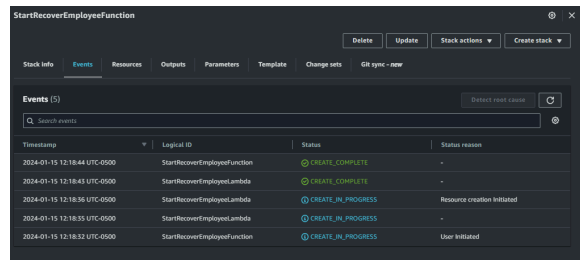
```

1  Resources:
2    # Función de Lambda
3    StartRecoverEmployeeLambda:
4      Type: AWS::Lambda::Function
5      Properties:
6        FunctionName: start-recover-employee
7        Role: arn:aws:iam::188912467164:role/LambdaStartRecoverEmployeeRole-BasicRole-SHOWefhL8Y0
8        Handler: index.handler
9        Runtime: nodejs16.x
10       Code:
11         S3Bucket: compumundo-lambda-code
12         S3Key: start-recover-employee.zip
13       Timeout: 29

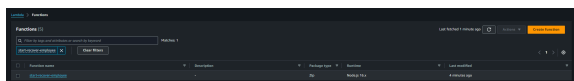
```

Creamos la plantilla .yml para la creación de la lambda start-recover-employee, asociando el role que se creó previamente y el archivo comprimido en formato .zip donde se encuentra el código fuente.

Cargamos en cloudformation la plantilla y la ejecutamos.



| Timestamp | Logical ID | Status | Status reason |
|------------------------------|------------------------------|--------------------|-----------------------------|
| 2024-01-15 12:18:44 UTC-0500 | StartRecoverEmployeeFunction | CREATE_COMPLETE | - |
| 2024-01-15 12:18:43 UTC-0500 | StartRecoverEmployeeLambda | CREATE_COMPLETE | - |
| 2024-01-15 12:18:36 UTC-0500 | StartRecoverEmployeeLambda | CREATE_IN_PROGRESS | Resource creation initiated |
| 2024-01-15 12:18:35 UTC-0500 | StartRecoverEmployeeLambda | CREATE_IN_PROGRESS | - |
| 2024-01-15 12:18:32 UTC-0500 | StartRecoverEmployeeFunction | CREATE_IN_PROGRESS | User initiated |



Validamos que se haya creado correctamente la función lambda.

Recursos: https://github.com/tatanvalencia/reto_pragma