Uso del SDK AWS en las pruebas automatizadas

El objetivo de esta guía es mostrar como podemos integrar el SDK AWS para java en las pruebas automatizadas para realizar acciones como son de consultas en DB, invocación de servicios aws entre otros, esto se lleva a cabo con el fin de realizar pruebas automatizadas mucho mas robustas que generen mas valor.

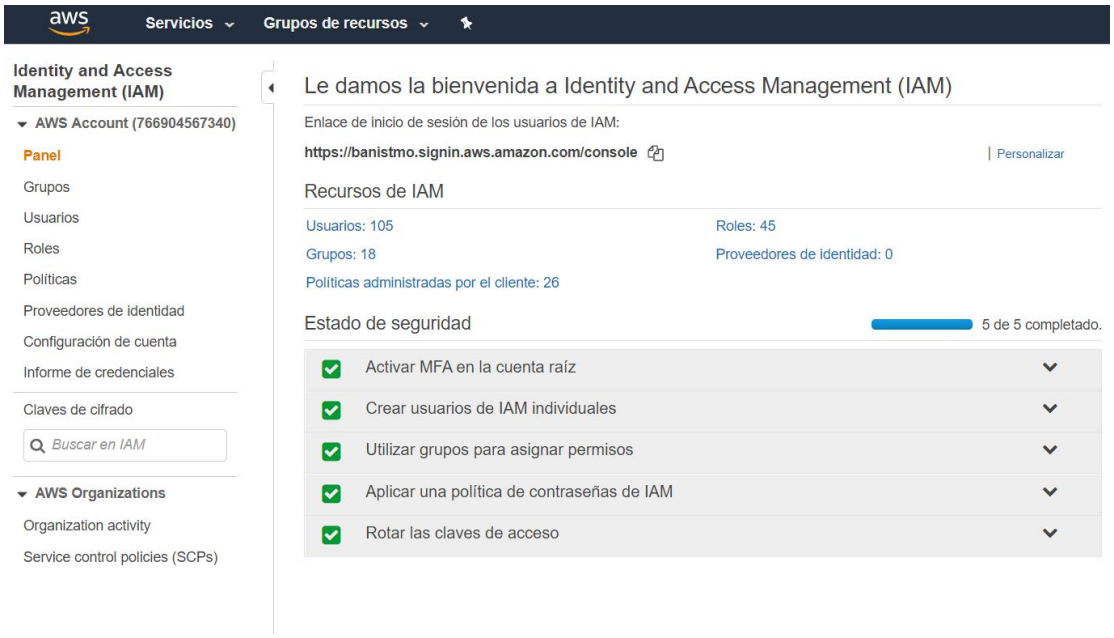
Para mostrar esto se realizara un prueba donde se inserta una persona a la DB para posteriormente invocar un servio llamado Cis.

El SDK permite eliminar la complejidad de la codificación, ya que proporciona API de Java para servicios de AWS, incluyendo Amazon S3, Amazon ECS, DynamoDB y AWS Lambda entre otros[1].

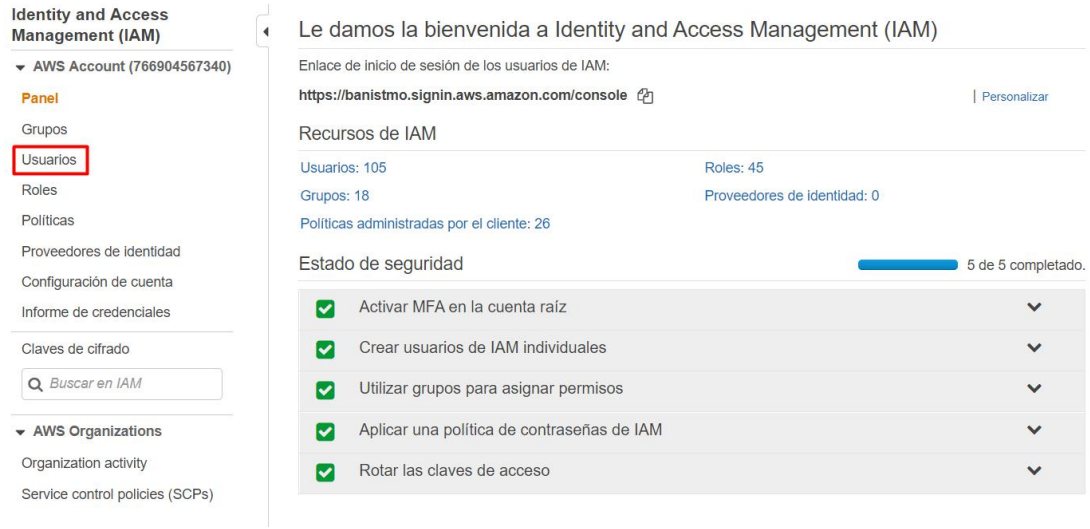
- **Obtener credenciales de AWS**

Las claves de acceso son las credenciales que le identifican en AWS y le permiten acceder programáticamente a los servicios y recursos de AWS. Las claves de acceso se pueden asociar con el "usuario raíz" de su cuenta de AWS o con los usuarios que cree con IAM[2].

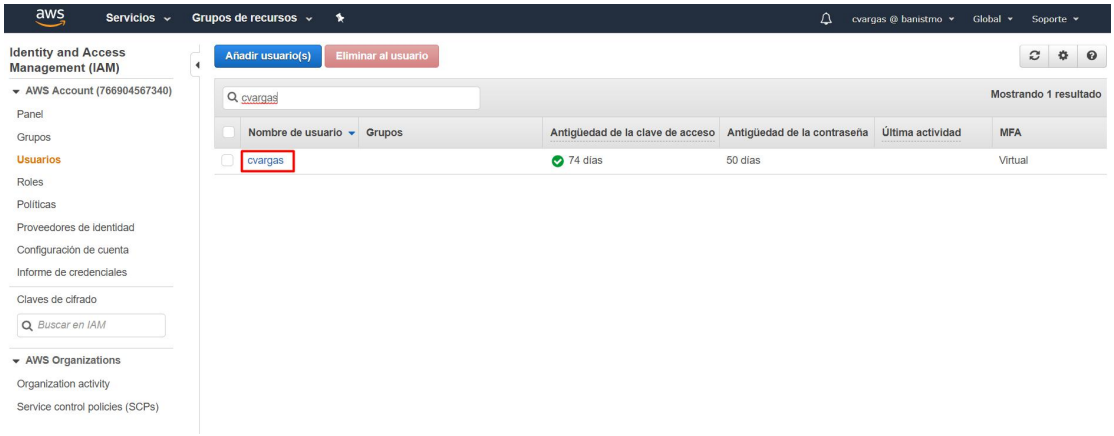
1. Abra la consola de IAM, en [https://console.aws.amazon.com/iam/.](https://console.aws.amazon.com/iam/)



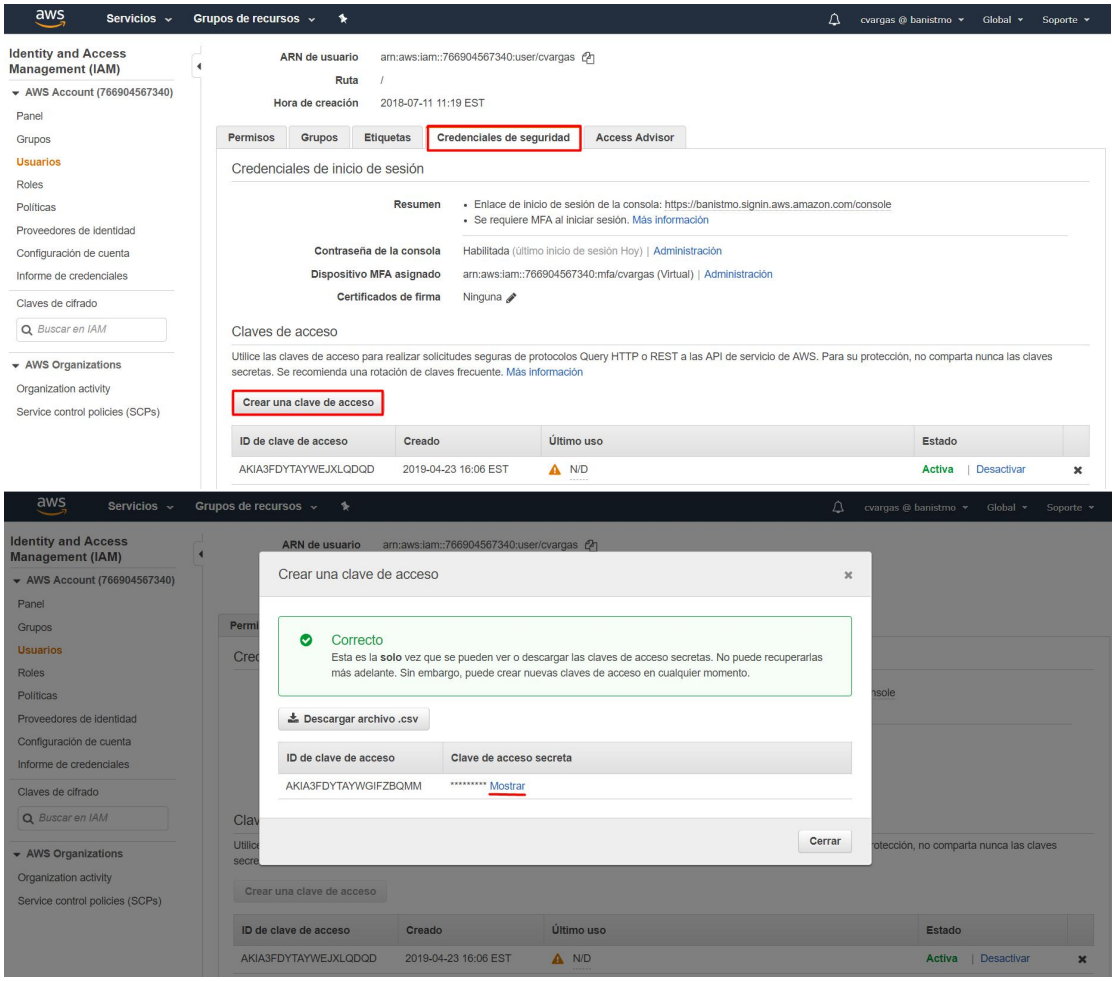
2. En el menú de navegación, elija Users (Usuarios).



3. Elija su nombre de usuario de IAM (no la casilla de verificación) para ver los detalles.



4. Elija la pestaña Security credentials (Credenciales de seguridad) y, a continuación, Create access key (Crear clave de acceso).



5. Para ver la nueva clave de acceso, elija Show (Mostrar). Las credenciales serán similares a las siguientes:

   ID de clave de acceso: AKIAIOSFODNN7EXAMPLE
   Clave de acceso secreta: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

Para descargar el par de claves, elija Download .csv file (Descargar archivo .csv). Almacene las claves en un lugar seguro.

Estas credenciales solo sirven para realizar ejecuciones locales, ya que cuando se ejecuta el test desde el pipeline este ya se encarga de realizar esta configuración, mas adelante se muestra como es este procedimiento.

- **Configuración del SDK para gradle**

- Importar el BOM como una dependencia habitual en la sección dependencies

```
dependencies {
    implementation 'com.amazonaws:aws-java-sdk-bom:1.11.228'
}
```

- Especifique los módulos del SDK que va a usar en la sección dependencies

```
dependencies {
    compile group: 'com.amazonaws', name: 'aws-java-sdk-dynamodb', version: '1.11.586'
    compile group: 'com.amazonaws', name: 'aws-java-sdk-lambda', version: '1.11.586'
    testCompile group: 'junit', name: 'junit', version: '4.11'
```

- Archivo build.gradle completo

```
group 'aws.test'
version '1.0-SNAPSHOT'

apply plugin: 'java'

sourceCompatibility = 1.8

repositories {
    mavenCentral()
}

dependencies {
    compile group: 'com.amazonaws', name: 'aws-java-sdk-dynamodb', version: '1.11.586'
    compile group: 'com.amazonaws', name: 'aws-java-sdk-lambda', version: '1.11.586'
    testCompile group: 'junit', name: 'junit', version: '4.11'
}
```

- **Creación del cliente de servicio**

Para realizar solicitudes a Amazon Web Services, primero se debe crear un objeto de cliente de servicio. La manera recomendada es utilizar el compilador de clientes de servicio.

```
private void createUser() throws Exception {

    ClassLoader classLoader = getClass().getClassLoader();
    String result =
IOUtils.toString(classLoader.getResourceAsStream("employee.json"))
;

    AWSCredentialsProvider credentials = new
AWSStaticCredentialsProvider(
            new
BasicAWSCredentials(System.getProperty("accessKey"),System.getPro
perty("secretKey")));

    AmazonDynamoDB client =
AmazonDynamoDBClientBuilder.standard().withRegion("us-east-1")
            .withCredentials(credentials).build();

    DynamoDB dynamoDB = new DynamoDB(client);

    Table tableFinal = dynamoDB.getTable("aws" +
System.getenv("GEOLOC") + "-" + System.getenv("STAGE")
            + System.getenv("ACCT") + "dbo" +
System.getenv("PROYECTO")+ "-employee");

    Item s = Item.fromJSON(result);
    tableFinal.putItem(s);

}
```

- ClassLoader carga dinámicamente la clase para obtener el recurso getClass() y getClasLoader() retorna el objeto classLoader

```
ClassLoader classLoader = getClass().getClassLoader();
```

- Devuelve una secuencia de entrada para leer el recurso *employee.json*

```
String result =
IOUtils.toString(classLoader.getResourceAsStream("employee.json"))
;
```

employee.json

```
{
  "contractType": "Temporal-Contrato Periodo Definido",
  "rol": "313",
  "agreementDate": "2019-03-07T16:34:49-05:00",
  "rolDescription": "Vendedor",
  "statusEmployee": "COMPLETADO",
  "birthdate": "1982-03-26T05:00:00.000Z",
  "maritalStatus": "S",
```

```
  "mobile": 43257686,
  "country": "PA",
  "province": "01",
  "district": "103",
  "corregimiento": "01013",
  "neighborhood": "El barrio obarrio",
  "street": "Calle 12 con calle 25",
  "house": "La casa verde de la esquina",
  "agree": true,
  "dniFilename": "8-744-369.jpg",
  "profession": "131",
  "nationality": "AE",
  "countryBorn": "AE",
  "isPep": true,
  "mailStatus": "ENVIADO",
  "urlLink":
"http://awsuseast1-sbxpllbucint-websites.s3-website-us-east-1.ama
zonaws.com/planilla/employee/123456",
  "checkControlListDate": "2018-06-25T17:35:56-05:00",
  "dni": "4-720-1385",
  "documentType": "C",
  "email1": "testingplanilla@gmail.com",
  "email2": "testingplanilla@gmail.com",
  "entryDate": "10/10/2017",
  "firstName": "Jose",
  "secondName": "David",
  "lastName": "Sierra",
  "secLastName": "Ramirez",
  "marriedName": "Toro",
  "layout": false,
  "layoutCIS": false,
  "payrollNumber": 23454323,
  "salary": 740,
  "gender": "M"
}
```

Este json contiene el colaborador que se va insertar en la tabla employee de DB, sin la propiedad CIS.

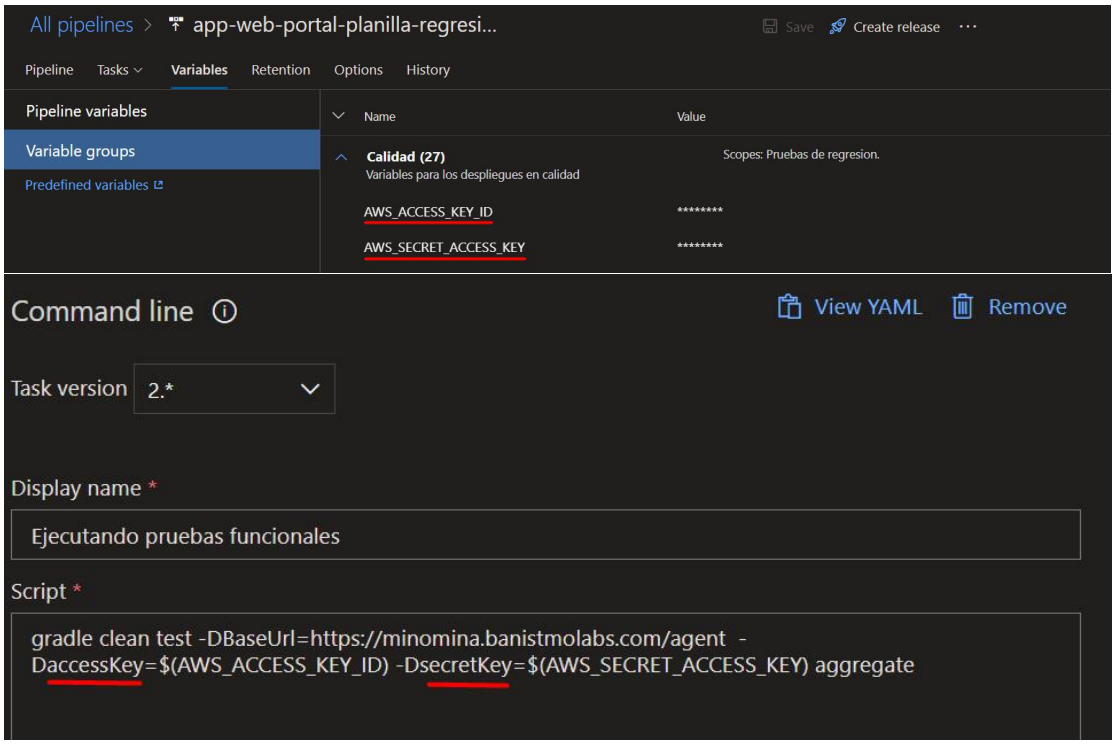
- Configuración de las credenciales de autenticación de AWS

```
AWSCredentialsProvider credentials = new
AWSStaticCredentialsProvider(
       new BasicAWSCredentials(System.getProperty("accessKey"),
System.getProperty("secretKey")));
```

Las credencias *accessKey* y *secretKey* se crean como propiedades en build.gradle para que sean ejecutadas por el pipeline donde se crean como variables.

```
test {
    systemProperty 'accessKey', System.getProperty('accessKey')
    systemProperty 'secretKey', System.getProperty('secretKey')
}
```

Configuración de las credenciales desde el pipeline



- Cada servicio de AWS tiene una interfaz de servicio con métodos para cada acción en la API de servicio. Por ejemplo, la interfaz de servicio de Amazon DynamoDB se llama AmazonDynamoDB. Cada interfaz de servicio cuenta con su compilador de clientes correspondiente, que puede utilizar para crear una implementación de la interfaz de servicio. La clase de compilador de clientes de DynamoDB se llama AmazonDynamoDBClientBuilder[3].

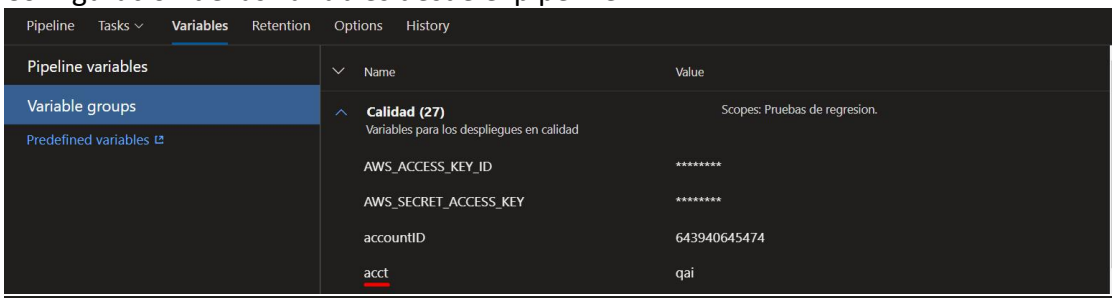
```
AmazonDynamoDB client =
AmazonDynamoDBClientBuilder.standard().withRegion("us-east-1")
       .withCredentials(credentials).build();
```

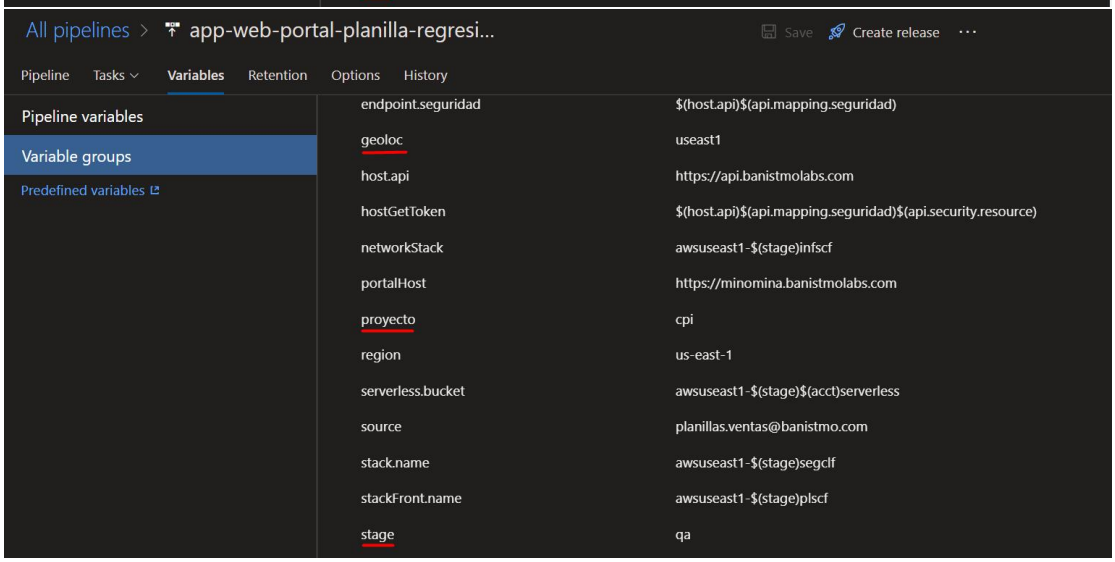
- Accediendo a la DB de dynamo con el cliente ya creado

```
DynamoDB dynamoDB = new DynamoDB(client);
Table tableFinal = dynamoDB.getTable("aws" +
System.getenv("GEOLOC")+ "-" + System.getenv("STAGE")
       + System.getenv("ACCT") + "dbo" +
System.getenv("PROYECTO")+"-employee");
```

Las variables de entorno GEOLOC, STAGE, ACCT, PROYECTO hacen parte del nombre de la tabla que se intenta acceder, estas también se encuentran configuradas desde el pipeline.

Configuración de las variables desde el pipeline





- Inserto el json a la tabla

```
Item s = Item.fromJSON(result);
tableFinal.putItem(s);
```

- **Consulta en DB**

```
public String consultDB() throws Exception{

    AWSCredentialsProvider credentials = new
AWSStaticCredentialsProvider(
          new
BasicAWSCredentials(System.getProperty("accessKey"),System.getPro
perty("secretKey")));

    AmazonDynamoDB client =
AmazonDynamoDBClientBuilder.standard().withRegion("us-east-1")
          .withCredentials(credentials).build();

    DynamoDB dynamoDB = new DynamoDB(client);
    //Table tableFinal =
dynamoDB.getTable("awsuseast1-devcpidbocpi-employee");
    Table tableFinal = dynamoDB.getTable("aws" +
System.getenv("GEOLOC") + "-" + System.getenv("STAGE")
          + System.getenv("ACCT") + "dbo" +
```

```
System.getenv("PROYECTO")+ "-employee");

    Item item = tableFinal.getItem("dni", "4-720-1385");

    return item.toJSONPretty();

}
```

- Consulta en DB al colaborador que se inserto con el json en el paso anterior

```
Item item = tableFinal.getItem("dni", "4-720-1385");
```

- **Invocación del servicio de Cis**

```
public void sendOk() throws Exception {

    createUser();
    System.out.println("Consulta ejecutada");
    String con = consultDB();

    System.out.println("Este es el empleado antes de consultar Cis");
    System.out.println(con);

    InvokeRequest invokeRequest;
    invokeRequest = new InvokeRequest()
            .withFunctionName("ms-fetch-cis-" +
System.getenv("STAGE")+ "-fetch");

    AWSCredentialsProvider credentials = new
AWSStaticCredentialsProvider(
            new
BasicAWSCredentials(System.getProperty("accessKey"),System.getPro
perty("secretKey")));

    AWSLambda awsLambda =
AWSLambdaClientBuilder.standard().withRegion("us-east-1").withCre
dentials(credentials)
            .build();

    InvokeResult invokeResult = null;
    invokeResult = awsLambda.invoke(invokeRequest);

   String con2 = consultDB();
   System.out.println("Este es el empleado despues de consultar Cis");
   System.out.println(con2);

    Assert.assertTrue(new
String(invokeResult.getPayload().array()).contains("Proceso
generado exitosamente"));
}
```

- Se crea el usuario

```
createUser();
```

- Se consulta a la persona antes de invocar la lambda

```
String con = consultDB();
System.out.println("Este es el empleado antes de consultar Cis");
System.out.println(con);
```

- Se crea un InvokeRequest que llamará a nuestra función Lambda

```
InvokeRequest invokeRequest;
invokeRequest = new InvokeRequest()
        .withFunctionName("ms-fetch-cis-" + System.getenv("STAGE")
+"-fetch");
```

- Se invoca la función Lambda

```
AWSLambda awsLambda =
AWSLambdaClientBuilder.standard().withRegion("us-east-1").withCre
dentials(credentials)
        .build();

InvokeResult invokeResult = null;
invokeResult = awsLambda.invoke(invokeRequest);
```

- Se consulta a la persona después de invocar la lambda

```
String con2 = consultDB();
System.out.println("Este es el empleado despues de consultar Cis");
System.out.println(con2);
```

- Verifica que la invocación fue realizada correctamente

```
Assert.assertTrue(new
String(invokeResult.getPayload().array()).contains("Proceso
generado exitosamente"));
```

- **Resultado de la Ejecución**

```
2019-07-04T20:07:36.6177865Z Consulta ejecutada
2019-07-04T20:07:37.0742955Z Este es el empleado antes de consultar Cis
2019-07-04T20:07:37.0756058Z {
2019-07-04T20:07:37.0757172Z "lastName" : "Sierra",
2019-07-04T20:07:37.0757955Z "country" : "PA",
2019-07-04T20:07:37.0758676Z "dniFilename" : "8-744-369.jpg",
2019-07-04T20:07:37.0759388Z "birthdate" : "1982-03-26T05:00:00.000Z",
2019-07-04T20:07:37.0760240Z "gender" : "M",
2019-07-04T20:07:37.0760855Z "documentType" : "C",
2019-07-04T20:07:37.0761629Z "agreementDate" : "2019-03-07T16:34:49-05:00",
2019-07-04T20:07:37.0762288Z "contractType" : "Temporal-Contrato Periodo Definido",
2019-07-04T20:07:37.0762923Z "rolDescription" : "Vendedor",
2019-07-04T20:07:37.0763673Z "countryBorn" : "AE",
2019-07-04T20:07:37.0764078Z "salary" : 740,
2019-07-04T20:07:37.0764218Z "house" : "La casa verde de la esquina",
2019-07-04T20:07:37.0764369Z "rol" : "313",
2019-07-04T20:07:37.0764664Z "layoutCIS" : false,
```

2019-07-04T20:07:37.0765127Z "email2" : "testingplanilla@gmail.com",
2019-07-04T20:07:37.0765348Z "email1" : "testingplanilla@gmail.com",
2019-07-04T20:07:37.0771844Z "province" : "01",
2019-07-04T20:07:37.0773007Z "street" : "Calle 12 con calle 25",
2019-07-04T20:07:37.0774326Z "urlLink" :
"http://awsuseast1-sbxpllbucint-websites.s3-website-us-east-1.amazonaws.com/planilla/employee/123456",
2019-07-04T20:07:37.0774719Z "dni" : "4-720-1385",
2019-07-04T20:07:37.0775305Z "secondName" : "David",
2019-07-04T20:07:37.0776423Z "profession" : "131",
2019-07-04T20:07:37.0777123Z "entryDate" : "10/10/2017",
2019-07-04T20:07:37.0777787Z "corregimiento" : "01013",
2019-07-04T20:07:37.0778315Z "mobile" : 43257686,
2019-07-04T20:07:37.0779692Z "mailStatus" : "ENVIADO",
2019-07-04T20:07:37.0781051Z "agree" : true,
2019-07-04T20:07:37.0781310Z "isPep" : true,
2019-07-04T20:07:37.0781465Z "layout" : false,
2019-07-04T20:07:37.0781711Z "firstName" : "Jose",
2019-07-04T20:07:37.0781826Z "nationality" : "AE",
2019-07-04T20:07:37.0782182Z "secLastName" : "Ramirez",
2019-07-04T20:07:37.0782264Z "district" : "103",
2019-07-04T20:07:37.0782465Z "payrollNumber" : 23454323,
2019-07-04T20:07:37.0782563Z "neighborhood" : "El barrio obarrio",
2019-07-04T20:07:37.0782686Z "checkControlListDate" : "2018-06-25T17:35:56-05:00",
2019-07-04T20:07:37.0782911Z "statusEmployee" : "COMPLETADO",
2019-07-04T20:07:37.0782985Z "marriedName" : "Toro",
2019-07-04T20:07:37.0783083Z "maritalStatus" : "S"
2019-07-04T20:07:37.0783231Z }
2019-07-04T20:07:54.6653381Z Este es el empleado despues de consultar Cis
2019-07-04T20:07:54.6656125Z {
2019-07-04T20:07:54.6656416Z "lastName" : "Sierra",
2019-07-04T20:07:54.6656699Z "country" : "PA",
2019-07-04T20:07:54.6656904Z "dniFilename" : "8-744-369.jpg",
2019-07-04T20:07:54.6657182Z "birthdate" : "1982-03-26T05:00:00.000Z",
2019-07-04T20:07:54.6657571Z "gender" : "M",
2019-07-04T20:07:54.6657866Z "documentType" : "C",
2019-07-04T20:07:54.6658083Z "agreementDate" : "2019-03-07T16:34:49-05:00",
2019-07-04T20:07:54.6658325Z "contractType" : "Temporal-Contrato Periodo Definido",
2019-07-04T20:07:54.6658694Z "rolDescription" : "Vendedor",
2019-07-04T20:07:54.6658890Z "countryBorn" : "AE",
2019-07-04T20:07:54.6659107Z "salary" : 740,
2019-07-04T20:07:54.6659365Z "house" : "La casa verde de la esquina",
2019-07-04T20:07:54.6659586Z "rol" : "313",
2019-07-04T20:07:54.6659910Z "layoutCIS" : false,
2019-07-04T20:07:54.6660119Z "email2" : "testingplanilla@gmail.com",
2019-07-04T20:07:54.6660344Z "email1" : "testingplanilla@gmail.com",
2019-07-04T20:07:54.6660557Z "province" : "01",
2019-07-04T20:07:54.6660774Z "street" : "Calle 12 con calle 25",
2019-07-04T20:07:54.6660987Z "urlLink" :
"http://awsuseast1-sbxpllbucint-websites.s3-website-us-east-1.amazonaws.com/planilla/employee/123456",
2019-07-04T20:07:54.6662736Z "dni" : "4-720-1385",
2019-07-04T20:07:54.6662953Z "secondName" : "David",
2019-07-04T20:07:54.6663146Z "profession" : "131",
2019-07-04T20:07:54.6663367Z "entryDate" : "10/10/2017",
2019-07-04T20:07:54.6663568Z "corregimiento" : "01013",
2019-07-04T20:07:54.6663801Z "mobile" : 43257686,
2019-07-04T20:07:54.6663998Z "mailStatus" : "ENVIADO",
2019-07-04T20:07:54.6664206Z "agree" : true,
2019-07-04T20:07:54.6664489Z "isPep" : true,
2019-07-04T20:07:54.6664678Z "CIS" : "49156",
2019-07-04T20:07:54.6664968Z "dniStatus" : "VALIDAR",
2019-07-04T20:07:54.6665165Z "layout" : false,
2019-07-04T20:07:54.6665374Z "firstName" : "Jose",
2019-07-04T20:07:54.6665575Z "nationality" : "AE",
2019-07-04T20:07:54.6665771Z "secLastName" : "Ramirez",
2019-07-04T20:07:54.6665988Z "district" : "103",
2019-07-04T20:07:54.6666222Z "payrollNumber" : 23454323,
2019-07-04T20:07:54.6666488Z "neighborhood" : "El barrio obarrio",
2019-07-04T20:07:54.6666701Z "checkControlListDate" : "2018-06-25T17:35:56-05:00",
2019-07-04T20:07:54.6666910Z "statusEmployee" : "COMPLETADO",
2019-07-04T20:07:54.6667143Z "marriedName" : "Toro",
2019-07-04T20:07:54.6667340Z "maritalStatus" : "S"
2019-07-04T20:07:54.6667549Z }

Como se puede evidenciar el colaborador se le ha consultado el cis.

- **Modo de ejecución en local y con el pipeline**

- Local:

gradle clean test -DBaseUrl=http://localhost:4200 -DaccessKey="AKIAJSY5HVG" -DsecretKey="CNy154kPDnNQVH9w" aggregate

Para esto se deben configurar las variables de entorno, una forma puede ser de la siguiente manera.

set STAGE=dev
set GEOLOC=useast1
set PROYECTO=cpi
set ACCT=cpi

- Pipeline

gradle clean test -DBaseUrl=http://minomina.banistmolabs.com -DaccessKey=$(AWS_ACCESS_KEY_ID) -DsecretKey=$(AWS_SECRET_ACCESS_KEY) aggregate

En este caso como vimos anteriormente las variables ya se encuentran configuradas en el pipeline.

Esta es una de las maneras de como se pueden realizar consultas a DB y invocación de servicios usando el SDK AWS desde las pruebas automatizadas.

- **Recursos**

1. Uso del SDK AWS
2. Obtener credenciales de AWS
3. Cliente de servicio
4. Toolkit para intellij