**Diseño y planificación**

En el desarrollo de las pruebas de aceptación se diseño el caso de prueba a automatizar, el cual es la emisión de un evento a la cola ibmmq para que el microservicio reaccione al evento y procese la información.  
  
para la creación de estas pruebas automatizadas se utilizo karate framework con java, librerías del banco para la conexión con aws y librerías de ibm para la conexión con la cola y la emisión de eventos.

Nos apoyamos con gradle y javascript para poder setear los valores que se requieren como archivos con los secretos de conexión y demás datos sensibles con el fin de poder setearlo desde el pipeline del microservicio.

Previo a la creación se realizo la investigación de como se pueden levantar los servicios de aws de forma local (localstack) y la cola ibm.

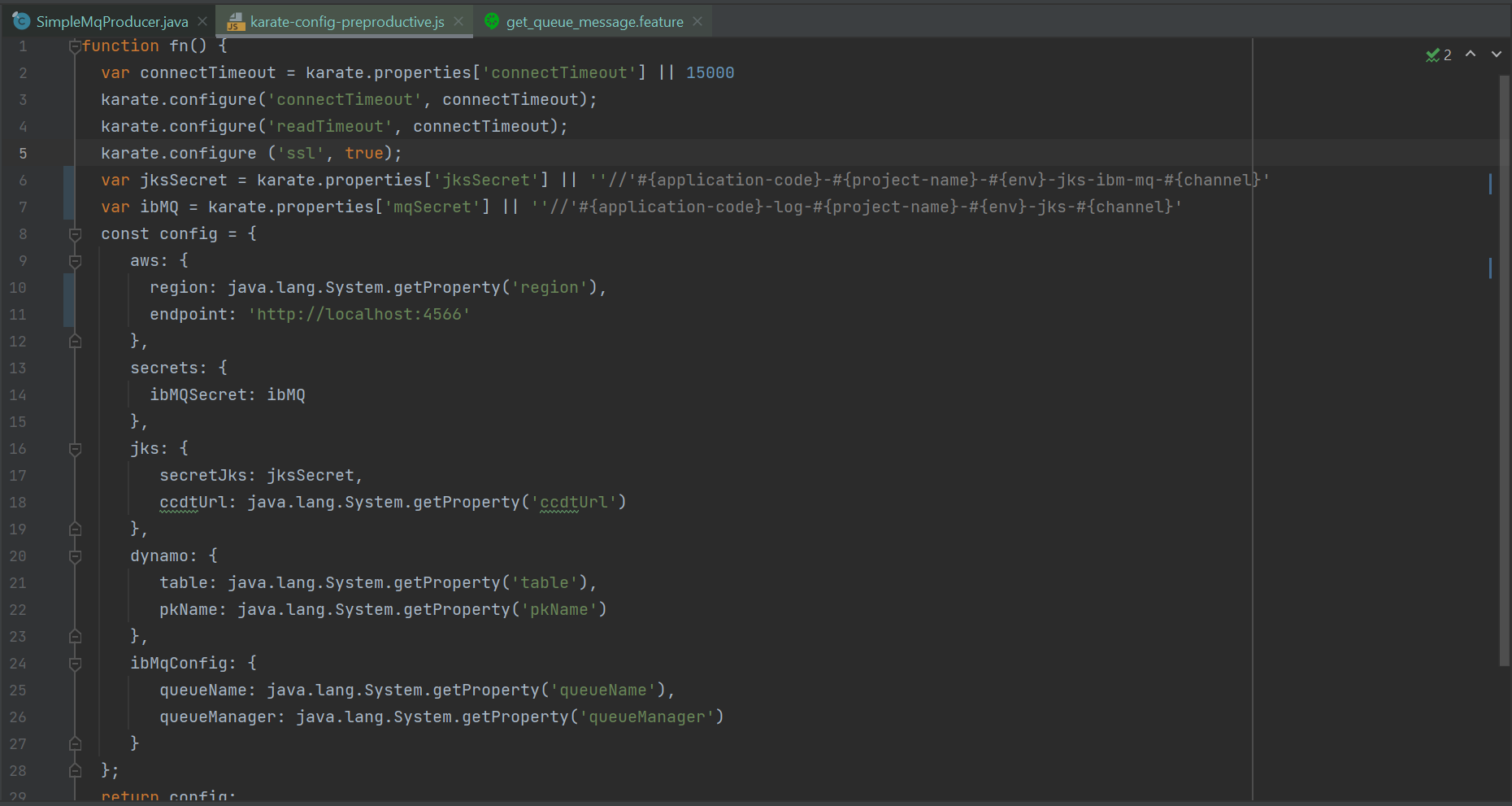
Para la conexión con AWS se utilizaron librerías de Bancolombia por tal motivo no se adjuntará evidencia del código pero se mostrará mediante los logs de la ejecución que se conecta de forma exitosa con los servicios del localstack y como se mapean los valores de secretos y propiedades de aws mediante gradle y karate-config.js:

**Desarrollo:**

La data de prueba se obtendrá de un bucketS3 pero para poder ejecutar localmente se obtendrá mediante un archivo .json el cual tiene dos objetos, el objeto de inserción y el objeto de eliminación.  
  
Texto

El contenido generado por IA puede ser incorrecto.

Karate-config.js (Archivo de configuración de karate)

Los valores de jksscret y ibMQ se mapean como vacíos para poder ejecutar el flujo de consumo de localstack  


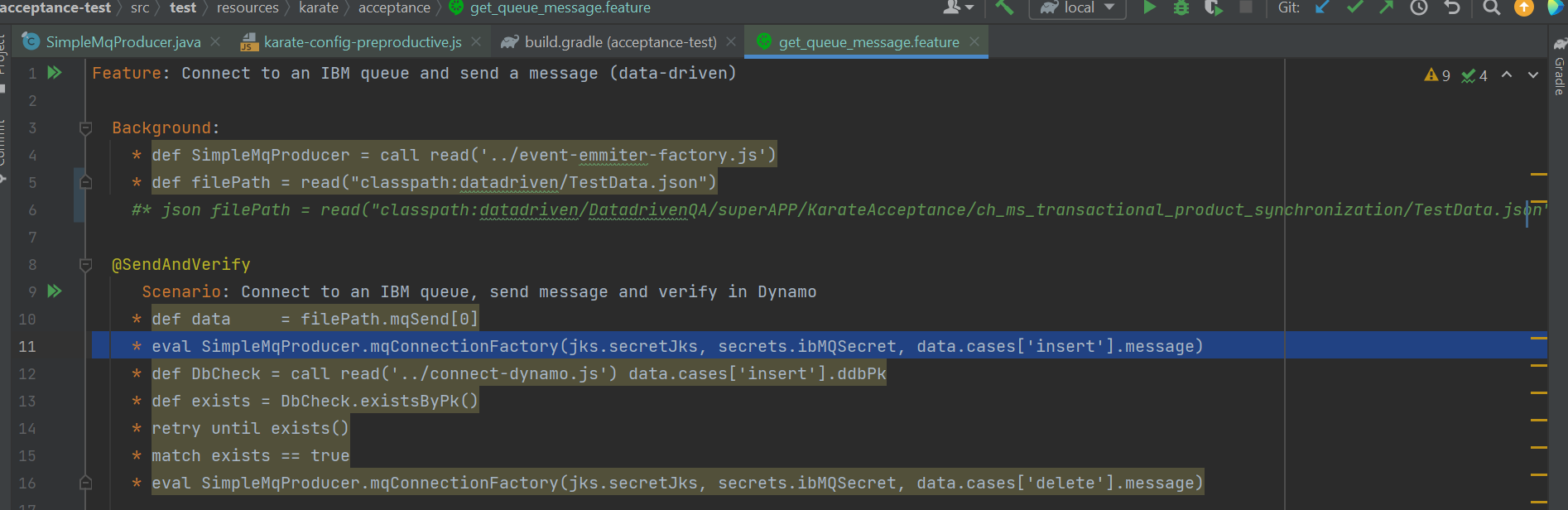
**Build.gradle**(gestor de dependencias y receptor de los valores de configuración por variables).

Se solicitan mediante el build gradle los valores requeridos para poder enviarlos por variable en el comando de ejecución de gradle

Texto

El contenido generado por IA puede ser incorrecto.

Archivo .feature(Archivo donde se construye el escenario de prueba en lenguaje Gherkin, también se indica el paso a paso de la ejecución del escenario).



En el archivo evento-emmiter-factory.js

Se realiza la conexión con aws y se crea la instancia de la case SimpleMqProducer, con los datos solicitados por constructor, con el fin de poder usar sus métodos en el archivo .feature.

Texto

El contenido generado por IA puede ser incorrecto.

La clase SimpleMqProducer conecta con la cola IBMMQ y se encarga de emitir los eventos, las variables **jksSecretArn** y **propertiesSecretArn** se inician vacías para poder ejecutar desde el flujo localTexto

El contenido generado por IA puede ser incorrecto.

Se realiza la conexión a la dynamoDB mediante connect-dynamo.js el cual retorna la instancia de la clase **DbCheck**

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

El método **client** se encarga de hacer la conexión con la dynamo, si el edpoint es null o “” significa que la conexión debe ser con la cuenta de aws de lo contrario se realiza con localstack.  
  
El método existByPk se encarga de realizar la consulta a la tabla de la dynamoDb con la llave compuesta, el método retorna un boolean si hay datos retorna true.

Texto

El contenido generado por IA puede ser incorrecto.

Se valida finalmente mediante una expresión del feature que que el valor retornado por el método sea verdadero:   


Y finalmente se envia un evento de eliminación reutilizando el método de emisión de eventos:  


Mapeo de la configuración de aws, se envia la propiedad de endpoint y de esta forma se interpreta que el consumo apunta a un localstack

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Para conectar con la cuenta AWS no se enviará endpoint con el fin de asegurar que se conectará a la cuenta directamente. Todo el código se ejecutará de igual forma.

**Ejecución**

**Ambiente local:**

Para poder ejecutar las pruebas automatizadas primero se necesita tener instalado aws cli, para poder crear un **localstack** y crear los componentes necesarios para subir el microservicio (cola ibmmq y dynamodb) de forma local. Previamente se descargó **WSL** para poder levantar contenedores **Docker**. Se configuro un contenedor docker con la siguiente imagen la cual permite levantar el **localstack**:   
version: '3.8'

services:

localstack:

container\_name: localstack-main

image: localstack/localstack:latest

ports:

- "4566:4566" # LocalStack Gateway

- "4510-4559:4510-4559" # external services port range

environment:

# LocalStack configuration

- DEBUG=1

- LAMBDA\_EXECUTOR=docker

- DOCKER\_HOST=unix:///var/run/docker.sock

- SERVICES=secretsmanager,s3,dynamodb

- DATA\_DIR=/tmp/localstack/data

- PORT\_WEB\_UI=8080

- PERSISTENCE=1

# AWS configuration

- AWS\_DEFAULT\_REGION=us-east-1

- AWS\_ACCESS\_KEY\_ID=test

- AWS\_SECRET\_ACCESS\_KEY=test

networks:

- localstack-network

# Opcional: AWS CLI para interactuar con LocalStack

#aws-cli:

# image: amazon/aws-cli:latest

#container\_name: aws-cli-localstack

# environment:

# - AWS\_ACCESS\_KEY\_ID=test

# - AWS\_SECRET\_ACCESS\_KEY=test

# - AWS\_DEFAULT\_REGION=us-east-1

# - AWS\_ENDPOINT\_URL=http://localstack:4566

# depends\_on:

# - localstack

# networks:

# - localstack-network

# profiles:

# - tools

# command: --version

networks:

localstack-network:

driver: bridge

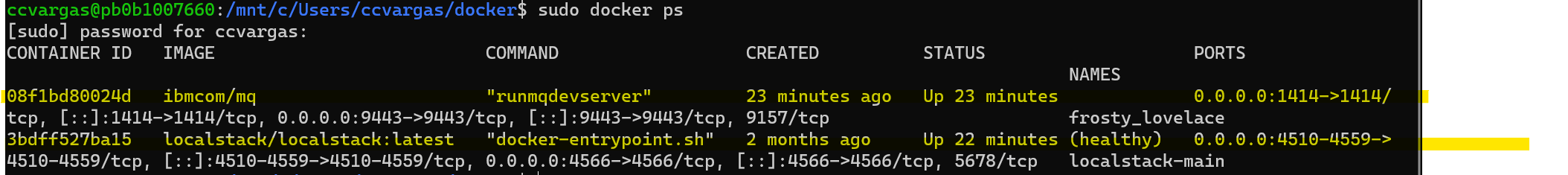
volumes:

localstack-data:

driver: local

posteriormente se crea la cola **ibmmq** como se indica en la siguiente documentación: [giovany79/ibm-mq: Proyecto que se implementa la escritura y la lectura en una cola MQ](https://github.com/giovany79/ibm-mq)

con los contenedores preparados y ejecutándose



Procedemos a crear los secretos de conexión a la cola

aws --endpoint-url=http://localhost:4566 --region us-east-1 secretsmanager create-secret --name nu001

4001-cgc-api-dev-keys-secret-mq --secret-string file://ibmmq.json --profile default

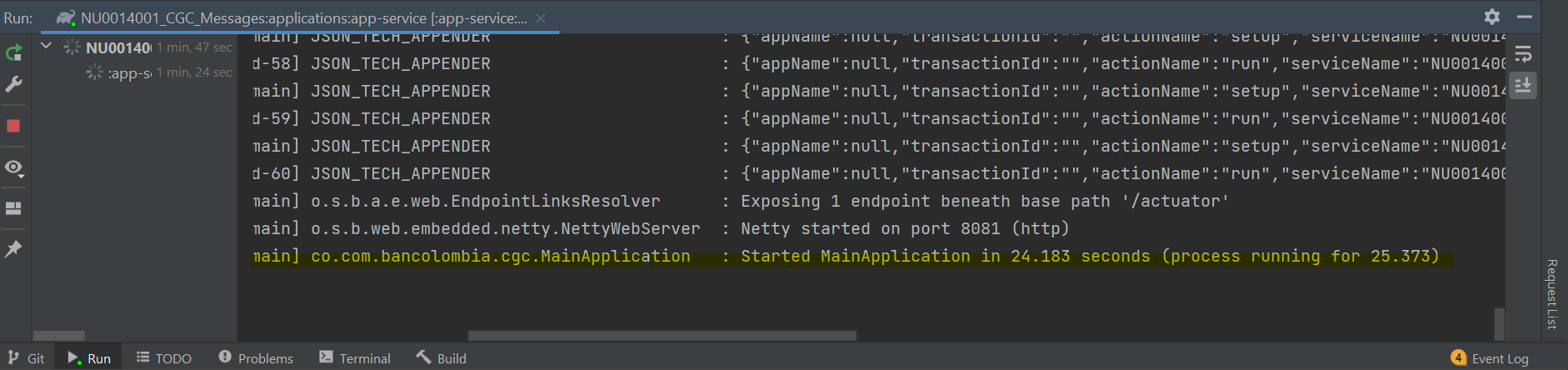
Se procesde a crear el **bucket S3**

aws --endpoint-url=http://localhost:4566 s3 mb s3://nu0014001-cgc-qa-s3-ibm

Se debe cargar el archivo nu0014001\_cgc\_ccdt\_qa.json en el bucket nu0014001-cgc-qa-s3-ibm dentro de la carpeta /MQ con el fin de que el proceso no falle y se pueda realizar la conexión a mq de manera correcta

aws --endpoint-url=http://localhost:4566 s3 cp nu0014001\_cgc\_ccdt\_qa.json s3://nu0014001-cgc-qa-s3-ibm/MQ/nu0014001\_cgc\_ccdt\_qa.json

se levanta el microservicio de forma local y se realiza la ejecución de la prueba de aceptación:



Se ejecuta las pruebas de aceptación con **gradle**:  
test --tests TestParallel -DccdtUrl=file:///C:/Users/ccvargas/banco/NU0621001\_super\_app\_p\_CH\_MR/ch\_ms\_transactional\_product\_syncronization/deployment/acceptance-test/src/test/resources/nu0621001\_app\_ccdt01.json -DqueueManager=\*CGCSGRP -DqueueName=CD\_MENSAPRODCGC\_WWW.QL.RES -DjksSecret=nu0014001-cgc-api-dev-keys-secret-mq -DmqSecret=nu0014001-cgc-api-dev-keys-secret-mq -Dregion=us-east-1 -Dtable=nu0014001-cgc-qa-dy-Account -DpkName=clientId -Dkarate.env=preproductive -i

Se ejecuta en **debug** con el fin de verificar los datos en la dynamodb:

**Recepción del mensaje de la cola en el microservicio:**

Imagen que contiene Texto

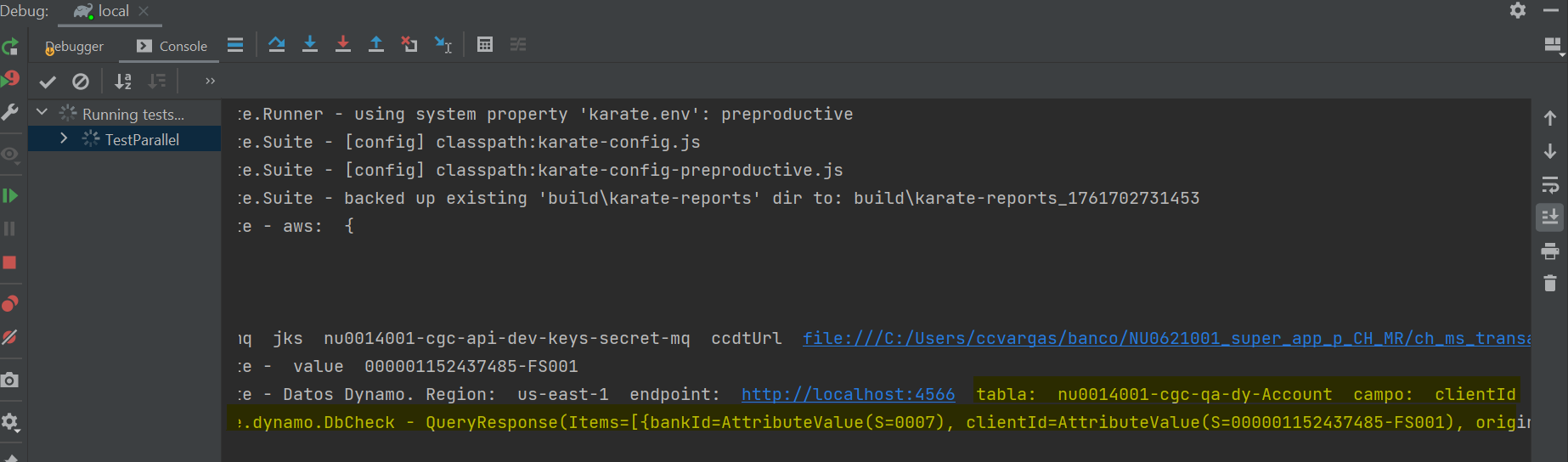
El contenido generado por IA puede ser incorrecto.

**Dynamo en localstack, se evidencia el registro**

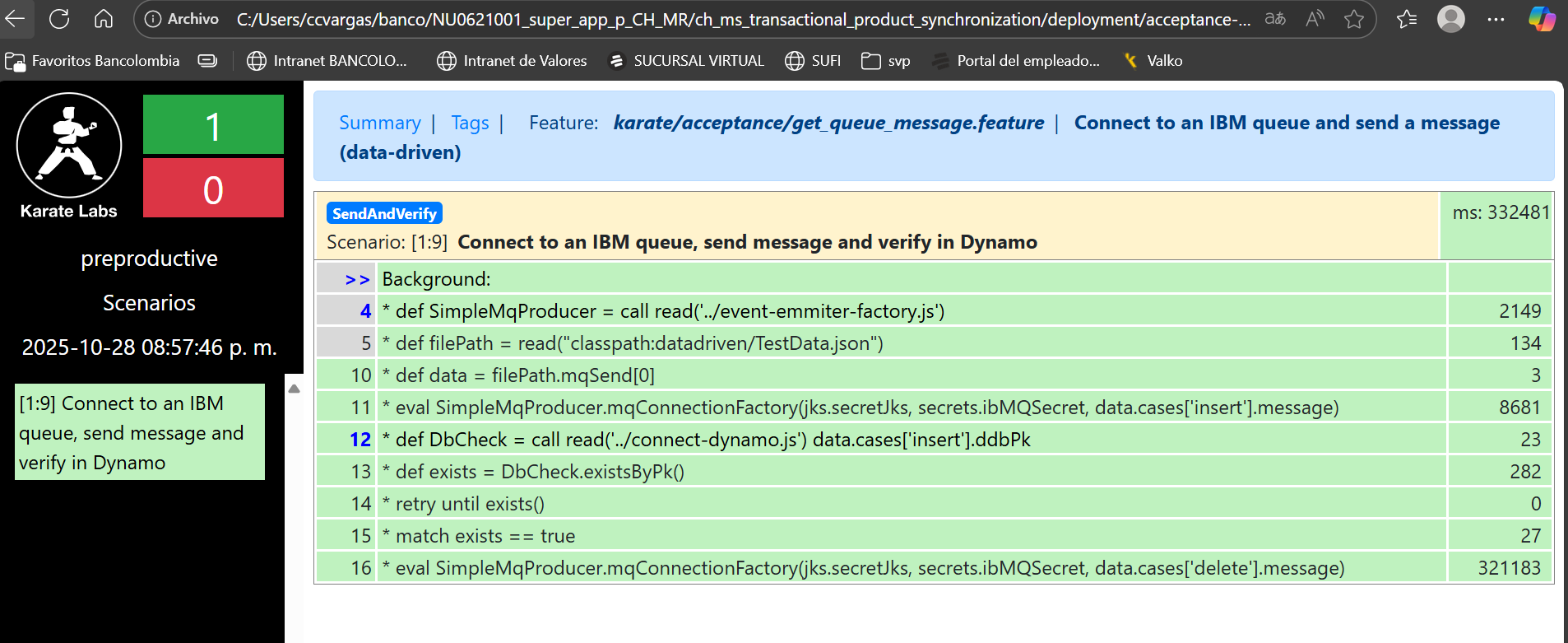
Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

**Logs del registro de la dynamo en las pruebas de aceptación:**

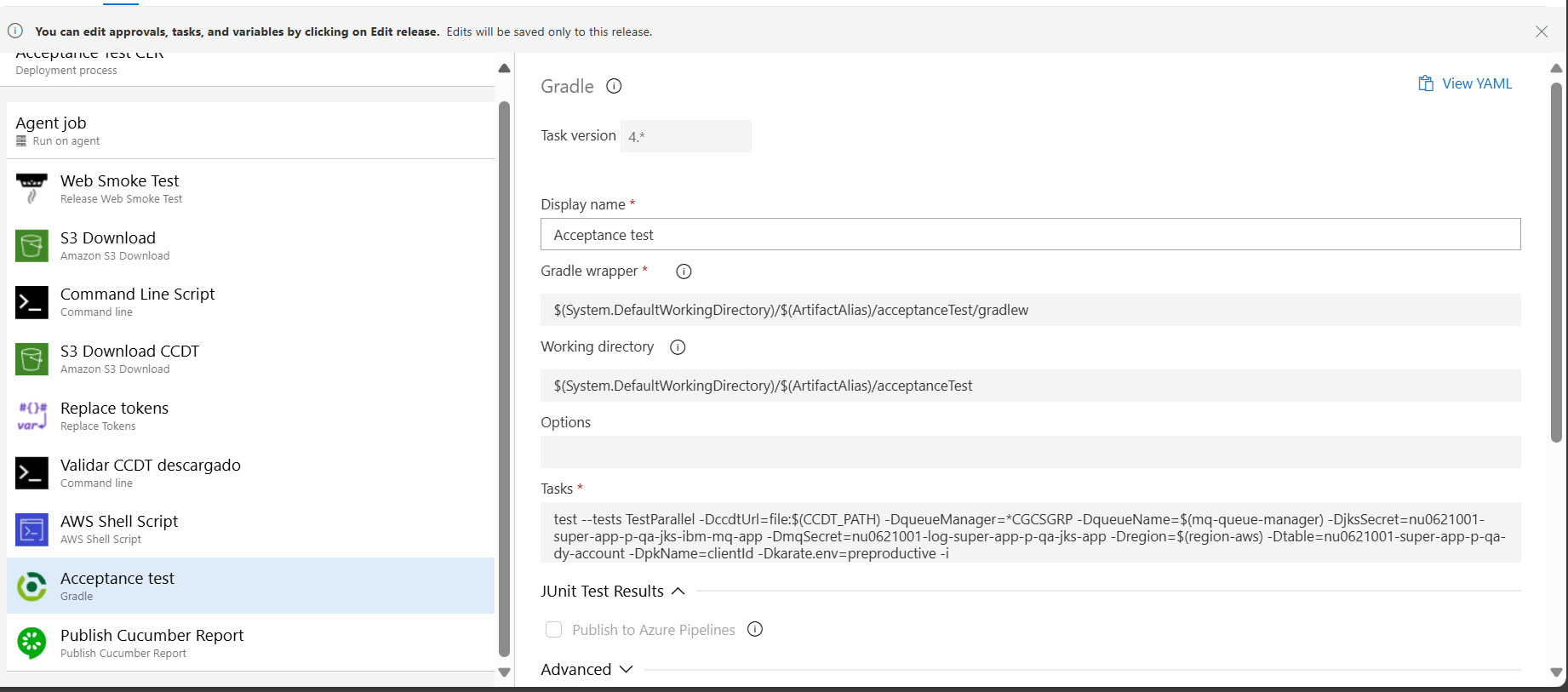


**Reporte** de las pruebas de aceptación ejecutadas exitosamente



**Ambiente de calidad desde el pipeline:**

Por protección de los datos no se mostrará la configuración de todas las tareas ya que en la tarea de S3 Download, CommandLineScript, S3 Download CCDT, Valdiar CCDT y AWS Shell Script hay configuración de credenciales y validaciones sensibles.

**La tarea AceptanTest ejecuta las pruebas de aceptación apuntando al ambiente preproductivo.  
**

**Reporte de la ejecución:  
Interfaz de usuario gráfica, Sitio web

El contenido generado por IA puede ser incorrecto.**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.**

**Resultado de la ejecución del stage en el pipelineInterfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.**

**Logs de la ejecución**

**Texto

El contenido generado por IA puede ser incorrecto.**