Paso 1: Crear el código fuente

En este paso, crea el código fuente que CodeBuild desea incluir en el segmento de salida. Este código fuente se compone de dos archivos de clases Java y un archivo Apache Maven Object Model (POM).

1. En un directorio vacío del equipo o la instancia local, cree esta estructura de directorios.

```
(root directory name)
    `-- src
        |-- main
        |     `-- java
        `-- test
                    `-- java
```

2. Con el editor de texto que desee, cree este archivo, asígnele el nombre `MessageUtil.java` y guárdelo en el directorio `src/main/java`.

```
public class MessageUtil {
  private String message;


  public MessageUtil(String message) {
    this.message = message;
  }


  public String printMessage() {
    System.out.println(message);
    return message;
```

```
  }


  public String salutationMessage() {
    message = "Hi!" + message;
    System.out.println(message);
    return message;
  }
    }
```

Este archivo de clases crea como resultado la cadena de caracteres que se la ha pasado. El constructor `MessageUtil` establece la cadena de caracteres. El método `printMessage` crea la salida. El método `salutationMessage` muestra `Hi!` seguido de la cadena de caracteres.

3. Cree este archivo, asígnele el nombre `TestMessageUtil.java` y guárdelo en el directorio `/src/test/java`.

```
import org.junit.Test;
import org.junit.Ignore;
import static org.junit.Assert.assertEquals;


public class TestMessageUtil {


  String message = "Robert";
  MessageUtil messageUtil = new MessageUtil(message);


  @Test
  public void testPrintMessage() {
```

```
    System.out.println("Inside testPrintMessage()");
    assertEquals(message,messageUtil.printMessage());
  }

  @Test
  public void testSalutationMessage() {
    System.out.println("Inside testSalutationMessage()");
    message = "Hi!" + "Robert";
    assertEquals(message,messageUtil.salutationMessage());
  }
     }
```

Este archivo de clases establece la variable `message` de la clase `MessageUtil` en `Robert`. A continuación, comprueba si la variable `message` se ha establecido correctamente comprobando si las cadenas `Robert` y `Hi!Robert` aparecen en la salida.

4. Cree este archivo, asígnele el nombre `pom.xml` y guárdelo en el directorio raíz (nivel superior).

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.example</groupId>
  <artifactId>messageUtil</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>
```

```
  <name>Message Utility Java Sample App</name>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
      </plugin>
    </plugins>
  </build>
</project>
```

Apache Maven utiliza las instrucciones de este archivo para convertir los archivos `MessageUtil.java` y `TestMessageUtil.java` en un archivo denominado `messageUtil-1.0.jar` y, a continuación, ejecuta las pruebas especificadas.

En este punto, la estructura de directorios debería ser similar a la siguiente.

```
(root directory name)
    |-- pom.xml
    `-- src
         |-- main
         |     `-- java
         |           `-- MessageUtil.java
         `-- test
               `-- java
                     `-- TestMessageUtil.java
```

Paso 2: Crear el archivo de especificación de compilación

En este paso, creará un archivo de especificación de compilación.
Una *especificación de compilación* es un conjunto de comandos de compilación y configuraciones relacionadas, en formato YAML, que se CodeBuild utiliza para ejecutar una compilación. Sin una especificación de compilación, CodeBuild no se puede convertir correctamente la entrada de compilación en el resultado de la compilación ni localizar el artefacto de salida de la compilación en el entorno de compilación para cargarlo en el depósito de salida.

Cree este archivo, asígnele el nombre `buildspec.yml` y guárdelo en el directorio raíz (nivel superior).

```
version: 0.2


phases:
  install:
    runtime-versions:
```

```
      java: corretto11
  pre_build:
    commands:
      - echo Nothing to do in the pre_build phase...
  build:
    commands:
      - echo Build started on `date`
      - mvn install
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  files:
    - target/messageUtil-1.0.jar
```

En esta declaración de especificación de compilación:

- `version` representa la versión del estándar de especificación de compilación que se va a usar. Esta declaración de especificación de compilación usa la última versión, `0.2`.
- `phases` representa las fases de compilación en las que puede indicar a CodeBuild que ejecute comandos. Estas fases de compilación se muestran aquí como `install`, `pre_build`, `build` y `post_build`. No puede cambiar los nombres de estas fases de compilación ni puede crear nombres de fases de compilación adicionales.

  En este ejemplo, durante la `build` fase, CodeBuild ejecuta el `mvn install` comando. Este comando indica a Apache Maven que compile, pruebe y empaquete los archivos de clases Java compilados en un artefacto de salida de la compilación. En aras de una mayor exhaustividad, se incluyen algunos comandos `echo` en cada fase de

compilación de este ejemplo. Cuando vea información de compilación detallada más adelante en este tutorial, la salida de estos comandos de `echo` puede ayudarlo a comprender mejor cómo CodeBuild ejecuta los comandos y en qué orden lo hace. (Aunque en este ejemplo se incluyen todas las fases de compilación, no es necesario que incluya una fase de compilación si no piensa ejecutar ningún comando durante esa fase). Para cada fase de compilación, CodeBuild ejecuta cada comando especificado, uno a la vez, en el orden indicado, de principio a fin.

- `artifacts`representa el conjunto de artefactos de salida de la compilación que se CodeBuild carga en el depósito de salida. `files`representa los archivos que se van a incluir en el resultado de la compilación. CodeBuild carga el único `messageUtil-1.0.jar` archivo que se encuentra en el directorio `target` relativo del entorno de compilación. El nombre del archivo `messageUtil-1.0.jar` y el nombre del directorio `target` se basan en la forma en que Apache Maven crea y almacena los artefactos de salida de la compilación para este ejemplo únicamente. En sus propias compilaciones, estos nombres de archivos y directorios son diferentes.

En este punto, la estructura de directorios debería ser similar a la siguiente.

```
(root directory name)
    |-- pom.xml
    |-- buildspec.yml
    `-- src
        |-- main
        |     `-- java
        |           `-- MessageUtil.java
        `-- test
              `-- java
                    `-- TestMessageUtil.java
```

Paso 3: Crear dos buckets de S3

Aunque puede usar un solo bucket para este tutorial, dos buckets hacen que sea más fácil ver de dónde proviene la entrada de compilación y hacia dónde va la salida de compilación.

- Uno de estos buckets (*bucket de entrada*) almacena la entrada de compilación. En este tutorial, el nombre de este depósito de entrada es `codebuild-`*`region-ID-account-ID`*`-input-bucket`, donde *`region-ID`* es la región de AWS del bucket y *`account-ID`* es su ID de cuenta de AWS.
- El otro bucket (*bucket de salida*) almacena la salida de la compilación. En este tutorial, el nombre de este bucket de salida es `codebuild-`*`region-ID-account-ID`*`-output-bucket`.

Si elige nombres diferentes para estos buckets, asegúrese de usarlos en este tutorial.

Estos dos buckets están en la misma región de AWS que sus compilaciones. Por ejemplo, si indicas CodeBuild que se ejecute una compilación en la región EE.UU. Este (Ohio), estos depósitos también deben estar en la región EE.UU. Este (Ohio).

Paso 4: Cargar el código fuente y el archivo buildspec

En este paso, añadirá el código fuente y el archivo de especificación de compilación al bucket de entrada.

Con la utilidad zip del sistema operativo, cree un archivo llamado `MessageUtil.zip` que incluya `MessageUtil.java`, `TestMessageUtil.java`, `pom.xml` y `buildspec.yml`.

La estructura de directorios del archivo `MessageUtil.zip` debe ser similar a la siguiente.

```
MessageUtil.zip

    |-- pom.xml
```

```
|-- buildspec.yml
`-- src
     |-- main
     |     `-- java
     |           `-- MessageUtil.java
     `-- test
           `-- java
                 `-- TestMessageUtil.java
```

Paso 5: Crear el proyecto de compilación

En este paso, creará un proyecto de compilación que usará AWS CodeBuild para ejecutar la compilación. Un *proyecto de compilación* incluye información sobre cómo ejecutar una compilación, incluido dónde obtener el código fuente, qué entorno de compilación se debe usar, qué comandos de compilación se deben ejecutar y dónde se debe almacenar el resultado de la compilación. Un *entorno de compilación* representa una combinación de sistema operativo, lenguaje de programación, tiempo de ejecución y herramientas que se CodeBuild utilizan para ejecutar una compilación. El entorno de compilación se expresa como una imagen de Docker. Para obtener más información, consulte Descripción general de Docker en la página de documentos de Docker.

Para este entorno de compilación, se indica CodeBuild que se utilice una imagen de Docker que contenga una versión del kit de desarrollo de Java (JDK) y Apache Maven.

Para crear el proyecto de compilación

1. Inicie sesión en la AWS Management Console y abra la consola de AWS CodeBuild en https://console.aws.amazon.com/codesuite/codebuild/home.

2. Usa el selector de AWS regiones para elegir una AWS región que sea compatible. CodeBuild Para obtener más información, consulte Puntos de conexión y cuotas de AWS CodeBuild en la *Referencia general de Amazon Web Services*.
3. Si se muestra una página de CodeBuild información, elija **Crear proyecto de construcción**. De lo contrario, en el panel de navegación, expanda **Compilar**, elija **Proyectos de compilación** y, a continuación, elija **Crear proyecto de compilación**.
4. En la página **Create build project (Crear proyecto de compilación)**, en **Project configuration (Configuración del proyecto)**, en **Project name (Nombre de proyecto)**, escriba un nombre para este proyecto de compilación (en este ejemplo, `codebuild-demo-project`). Los nombres de los proyectos de compilación debe ser únicos en cada cuenta de AWS. Si elige otro nombre, asegúrese de utilizarlo durante todo el tutorial.

5. En **Fuente**, como **Proveedor de código fuente**, elija **Amazon S3**.
6. En **Bucket**, elija **codebuild-*region-ID*-*account-ID*-input-bucket**.
7. En **S3 object key (Clave de objeto de S3)**, escriba `MessageUtil.zip`.
8. En **Environment (Entorno)**, para **Environment image (Imagen de entorno)**, deje **Managed image (Imagen administrada)** seleccionado.
9. En **Operating system (Sistema operativo)**, elija **Amazon Linux 2**.
10. En **Runtime(s) (Tiempo de ejecución)**, elija **Standard (Estándar)**.
11. En **Imagen**, elija **aws/codebuild/amazonlinux2-x86_64-standard:4.0**.
12. En **Service role (Rol de servicio)**, deje la opción **New service role (Nuevo rol de servicio)** seleccionada y no haga ningún cambio en **Role name (Nombre de rol)**.
13. En **Buildspec**, deje **Use a buildspec file (Usar un archivo buildspec)** seleccionado.
14. En **Artefactos**, como **Tipo**, seleccione **Amazon S3**.
15. En **Nombre de bucket**, elija **codebuild-*regon-ID*-*account-ID*-output-bucket**.
16. Deje **Name (Nombre)** y **Path (Ruta)** en blanco.
17. Elija **Crear el proyecto de compilación**.

Paso 6: Ejecutar la compilación

En este paso, indicará a AWS CodeBuild que ejecute la compilación con la configuración del proyecto de compilación.

Para ejecutar la compilación

1. Abra la consola de AWS CodeBuild en https://console.aws.amazon.com/codesuite/codebuild/home.
2. En el panel de navegación, elija **Proyectos de compilación**.
3. En la lista de proyectos de construcción, elija y **codebuild-demo-project**, a continuación, elija **Iniciar compilación**. La compilación comienza inmediatamente.

Paso 7: Ver información resumida sobre la compilación

En este paso, verá información resumida sobre el estado de la compilación.

Para ver la información resumida de la compilación

1. Si <build-ID>no se muestra la página **codebuild-demo-project:**, en la barra de navegación, seleccione **Crear historial**. A continuación, en la lista de proyectos de construcción, para **Proyecto**, elija el enlace **Construir y ejecutar** para **codebuild-demo-project**. Debe haber un solo enlace coincidente. (Si ya ha completado este tutorial antes, elija el vínculo con el valor más reciente en la columna **Completed (Completado)**).
2. En la página **Estado de compilación**, en **Detalles de fase**, se deben mostrar las fases de compilación siguientes como **Con éxito** en la columna **Estado**:
    - **SUBMITTED**
    - **QUEUED**
    - **PROVISIONING**
    - **DOWNLOAD_SOURCE**
    - **INSTALL**

- **PRE_BUILD**
- **BUILD**
- **POST_BUILD**
- **UPLOAD_ARTIFACTS**
- **FINALIZING**
- **COMPLETED**

En **Build Status (Estado de la compilación)**, debería mostrarse **Succeeded (Realizado correctamente)**.

Si en su lugar aparece **In Progress (En curso)** elija el botón de actualizar.

3. Junto a cada fase de compilación, el valor **Duration (Duración)** indica cuánto tiempo ha tardado la fase de compilación. El valor **End time (Hora de finalización)** indica que esa fase de compilación ha terminado.

Paso 8: Ver información detallada sobre la compilación

En este paso, verás información detallada sobre tus CloudWatch registros de compilación.

Para ver información detallada sobre la compilación

1. Con la página de detalles de la compilación mostrada del paso anterior, se muestran las últimas 10,000 líneas del log de compilación en **Build logs**. Para ver el registro de compilación completo en CloudWatch Logs, selecciona el enlace **Ver todo el registro**.
2. En el flujo de registro de CloudWatch registros, puedes buscar los eventos del registro. De forma predeterminada, solo se muestra el último conjunto de eventos de log. Para ver eventos de log anteriores, desplácese hasta el principio de la lista.
3. En este tutorial, la mayoría de los eventos de registro contienen información detallada sobre la descarga e instalación de archivos de dependencia de compilación de CodeBuild en su entorno de compilación, lo que probablemente no le interese. Puede usar el cuadro **Filter events** para reducir la información que se muestra. Por ejemplo, si escribe `"[INFO]"` en el cuadro **Filter events (Filtrar eventos)**, solo se muestran los eventos que contienen `[INFO]`. Para obtener más

información, consulte Sintaxis de filtros y patrones en la *Guía del CloudWatch usuario de Amazon*.

Paso 9: Obtener el artefacto de salida de la compilación

En este paso, obtendrá el `messageUtil-1.0.jar` archivo que se CodeBuild creó y se cargó en el depósito de salida.

Puede utilizar la CodeBuild consola o la consola Amazon S3 para completar este paso.

Para obtener el artefacto de salida de la compilación (consola de AWS CodeBuild)

1. Con la CodeBuild consola aún abierta y la página de detalles de la compilación del paso anterior, selecciona la pestaña **Detalles de la construcción** y desplázate hacia abajo hasta la sección **Artefactos**.
   nota

   Si la página de detalles de la compilación no se muestra, en la barra de navegación, elija **Historial de compilación** y, a continuación, elija el enlace **Ejecución de la compilación**.
2. El enlace a la carpeta Amazon S3 se encuentra en la **Ubicación de carga de artefactos**. Este enlace abre la carpeta de Amazon S3 donde se encuentra el archivo de artefactos de salida de compilación de `messageUtil-1.0.jar`.

Para obtener el artefacto de salida de la compilación (consola de Amazon S3)

1. Abra la consola de Amazon S3 en https://console.aws.amazon.com/s3.
2. Abra `codebuild-`*`region-ID`*`-`*`account-ID`*`-output-bucket`.
3. Abra la carpeta `codebuild-demo-project`.
4. Abra la carpeta `target`, donde se encuentra el archivo de artefactos de salida de compilación `messageUtil-1.0.jar`.