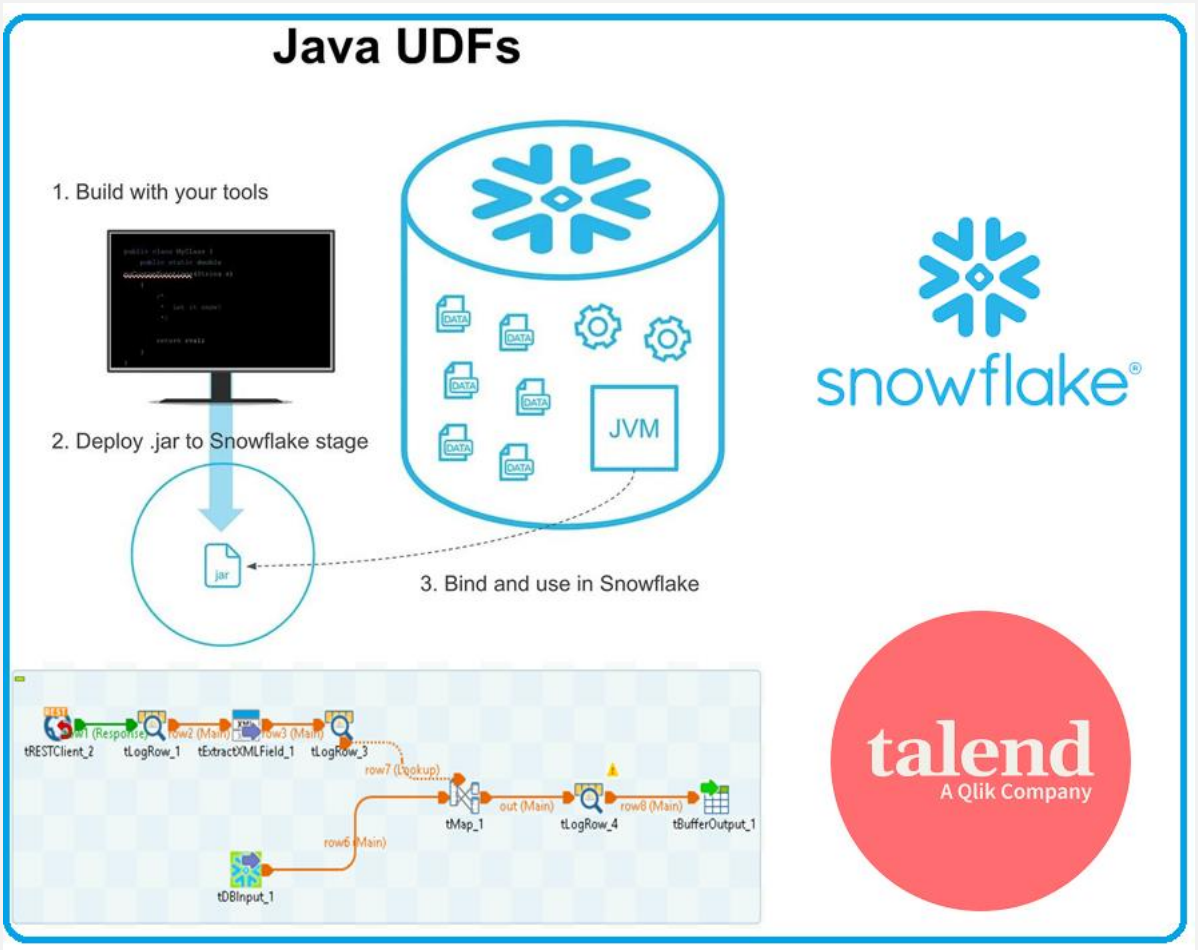
INTEGRACIÓN DE SNOWFLAKE Y TALEND

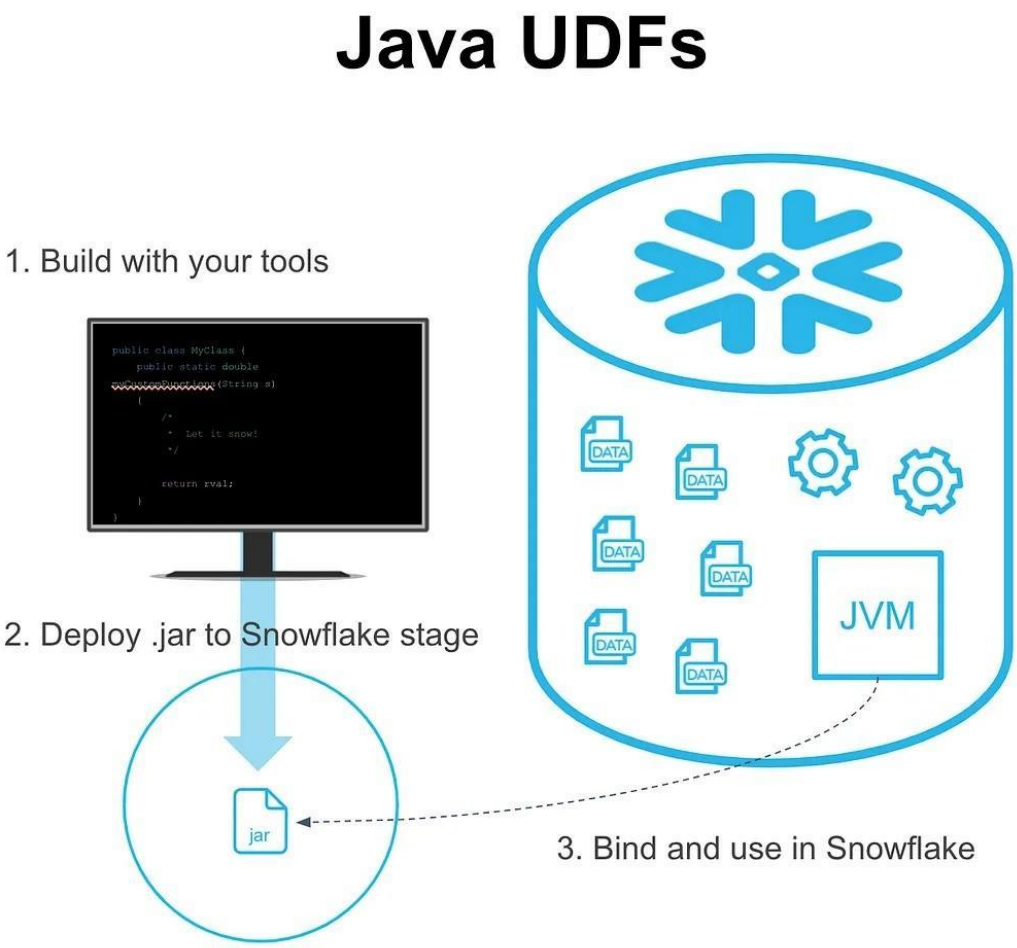
Cómo incorporar jobs de Talend como funciones UDF en Snowflake



ÍNDICE

INTRODUCCIÓN

En este paper se explica el caso de uso del desarrollo de funciones definidas por el usuario (UDF) trabajando con Talend y Snowflake. Esta forma de trabajar permite ampliar las posibilidades de Snowflake, haciendo posible utilizar características de Talend en esta plataforma.



En la imagen anterior, se puede observar un ejemplo esquemático de cómo funciona este proceso. Cabe destacar que Snowflake permite tratar con diferentes tipos de fuentes de datos, ya sean estructuradas, semi estructuradas o no estructuradas, lo cual da un amplio abanico de posibilidades al desarrollador permitiéndole integrar y procesar datos de diversas fuentes en un entorno unificado.

El principal objetivo es envolver trabajos de Talend en una clase de Java que actúe como un "wrapper" y permita que el código de Talend sea ejecutado dentro de Snowflake como una UDF. Para lograrlo, el flujo incluye los siguientes pasos fundamentales:

1. **Adaptación del trabajo de Talend**: Un trabajo diseñado en Talend Studio se adapta para ser compatible con el entorno Snowflake mediante el uso de una clase "wrapper". Esta clase se encarga de definir la lógica necesaria para recibir parámetros, ejecutar el trabajo, y devolver resultados o mensajes de error.
2. **Generación de artefactos Maven**: Utilizando Maven, se empaqueta el trabajo en un archivo JAR que contiene todos los elementos necesarios para la ejecución de la UDF, incluyendo todas las dependencias, configuración y lógica de ejecución.
3. **Publicación del artefacto en Snowflake**: El archivo JAR se sube a un "stage" de Snowflake (almacenamiento interno) y se registra como una UDF mediante comandos SQL.
4. **Pruebas y validación**: Finalmente, la UDF registrada se prueba ejecutando una sencilla consulta SQL en Snowflake, confirmando que esta devuelve lo que se espera sin dar lugar a errores.

REQUISITOS TÉCNICOS

Para implementar esta solución, se requieren los siguientes componentes y conocimientos:

- Talend Data Integration Platform Studio en su versión 8.0.1 o superior.
- Una cuenta de Snowflake, con los permisos necesarios para poder crear UDFs
- Herramientas de desarrollo como Maven y repositorios Nexus configurados para manejar artefactos de Talend.
- Conocimientos en Java y SQL.

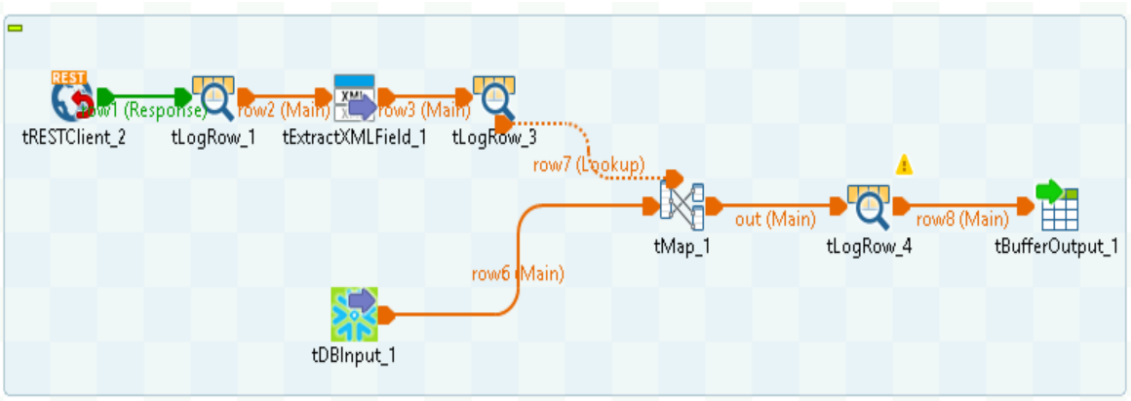
CONTEXTO Y CONFIGURACIÓN INICIAL

El proceso comienza con la configuración de un proyecto base que incluye los recursos esenciales:

- **Clase Wrapper**: Contiene la lógica que permite ejecutar trabajos de Talend como UDFs. Esta clase procesa parámetros de entrada, ejecuta el trabajo y devuelve resultados.
- **Archivos de Maven y Nexus**: Configurados para automatizar la generación de artefactos y su integración con Talend.

CASO DE USO: EJECUCIÓN DE UNA TRANSFORMACIÓN COMPLEJA

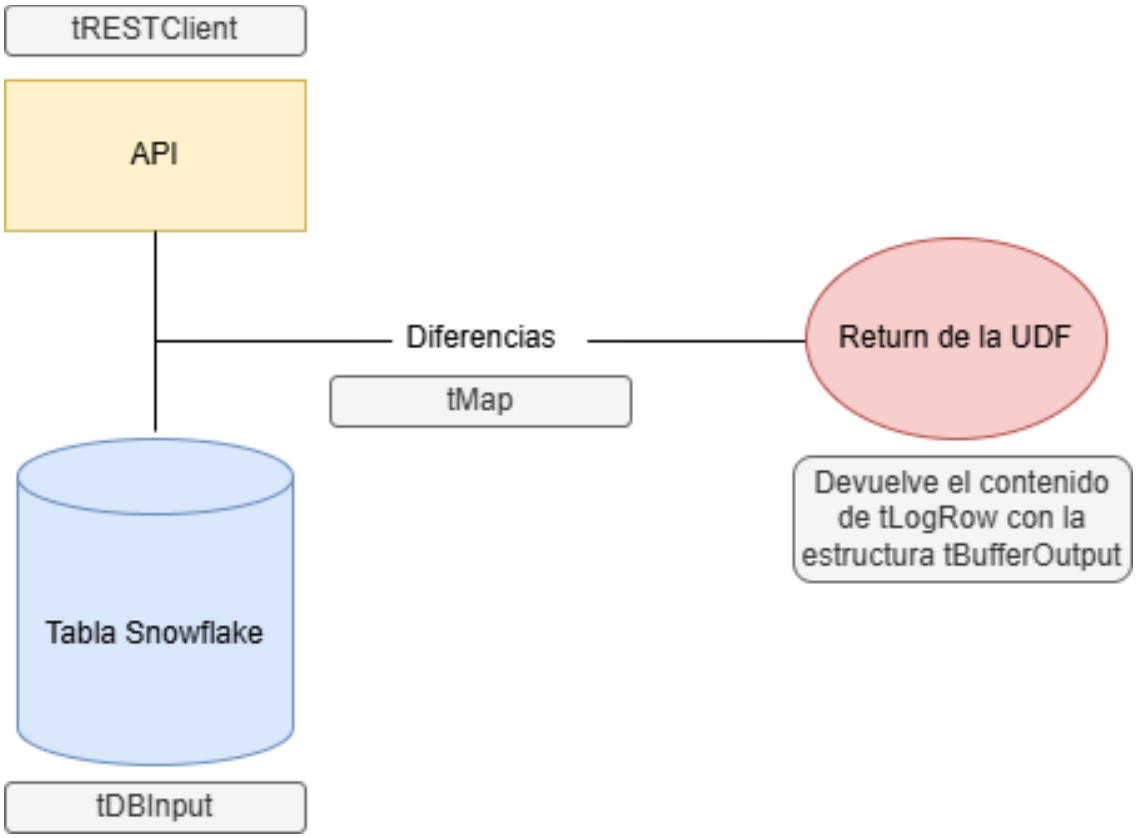
En este caso de prueba facilitado por Talend se utiliza un job sencillo con el nombre de HelloWorld. El Job "HelloWorld" tiene un propósito básico, diseñado como una prueba conceptual para demostrar cómo un trabajo de Talend puede ejecutarse como una User-Defined Function (UDF) en Snowflake.



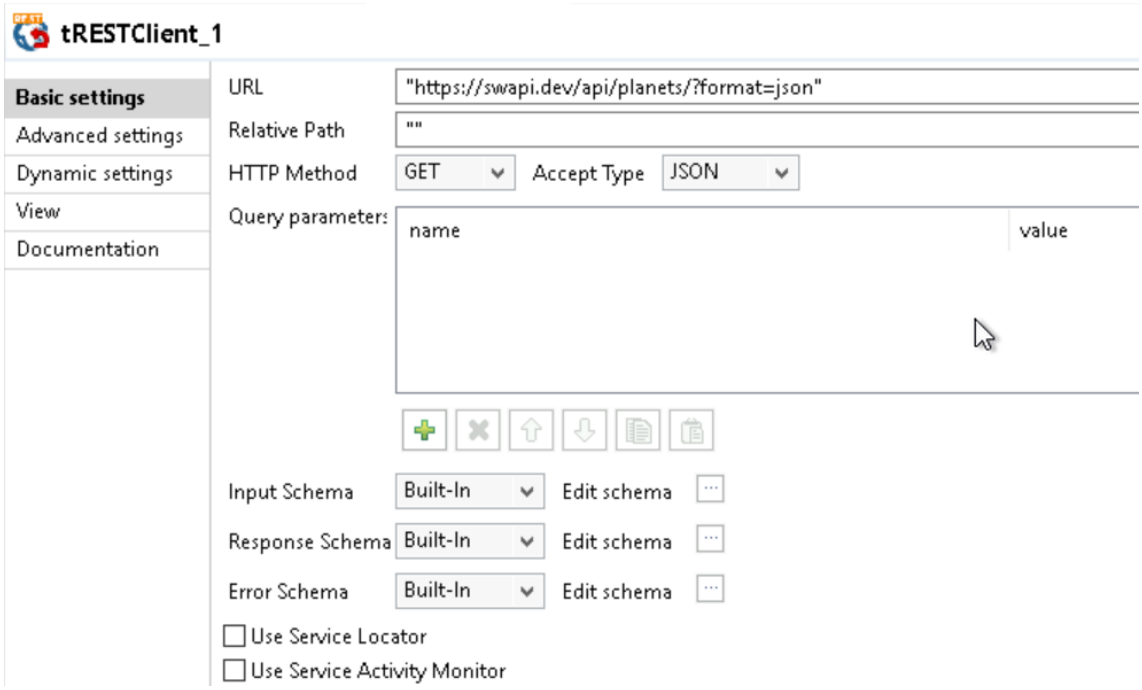
Un escenario real podría ser el de un job como el de la imagen anterior, que hace referencia al siguiente escenario: Una empresa quiere automatizar la comparación entre los datos almacenados en Snowflake y los obtenidos de un API externo. El objetivo no es registrar las discrepancias en una tabla, sino mostrarlas directamente en la consola, donde pueden ser revisadas o redirigidas a logs para su análisis.

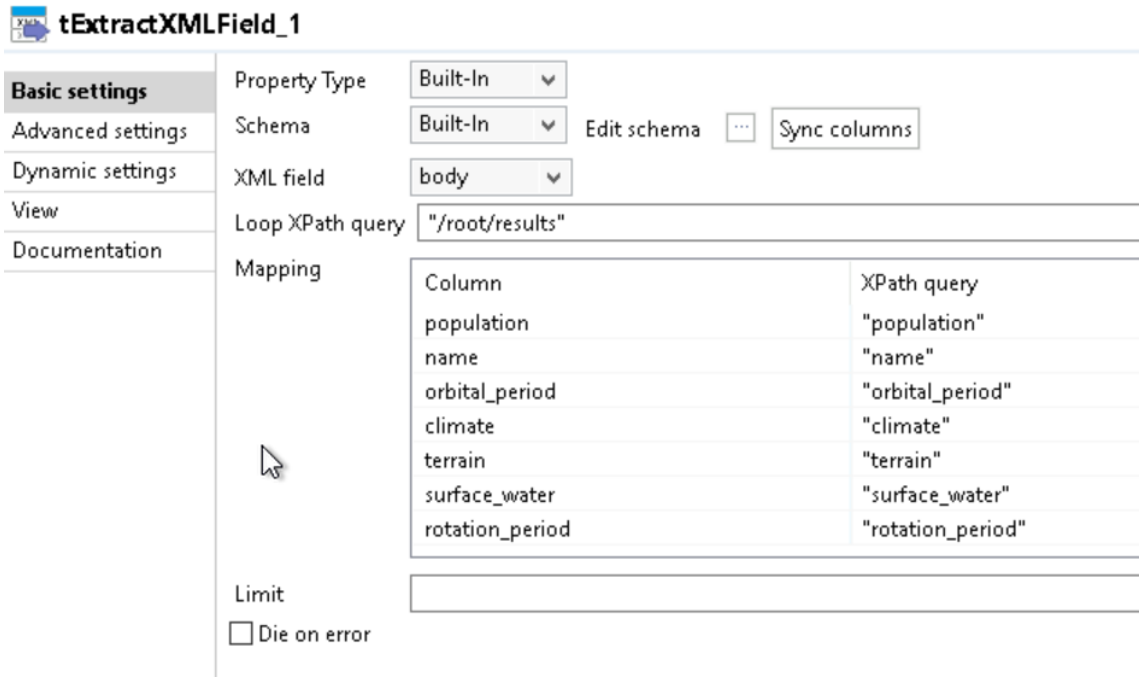
En este caso, el API escogida es una que ofrece datos de Star Wars sin ningún tipo de identificación requerida, simplemente se hacen las peticiones necesarias y esta envía los datos como respuesta. La tabla de Snowflake (con el nombre Planets) contiene datos de los Planetas presentes en la saga de películas (nombre y población). El job, a través de un tMap, compara si la población almacenada en la tabla de Snowflake difiere de la que aparece en los datos del API y lo imprime por pantalla.

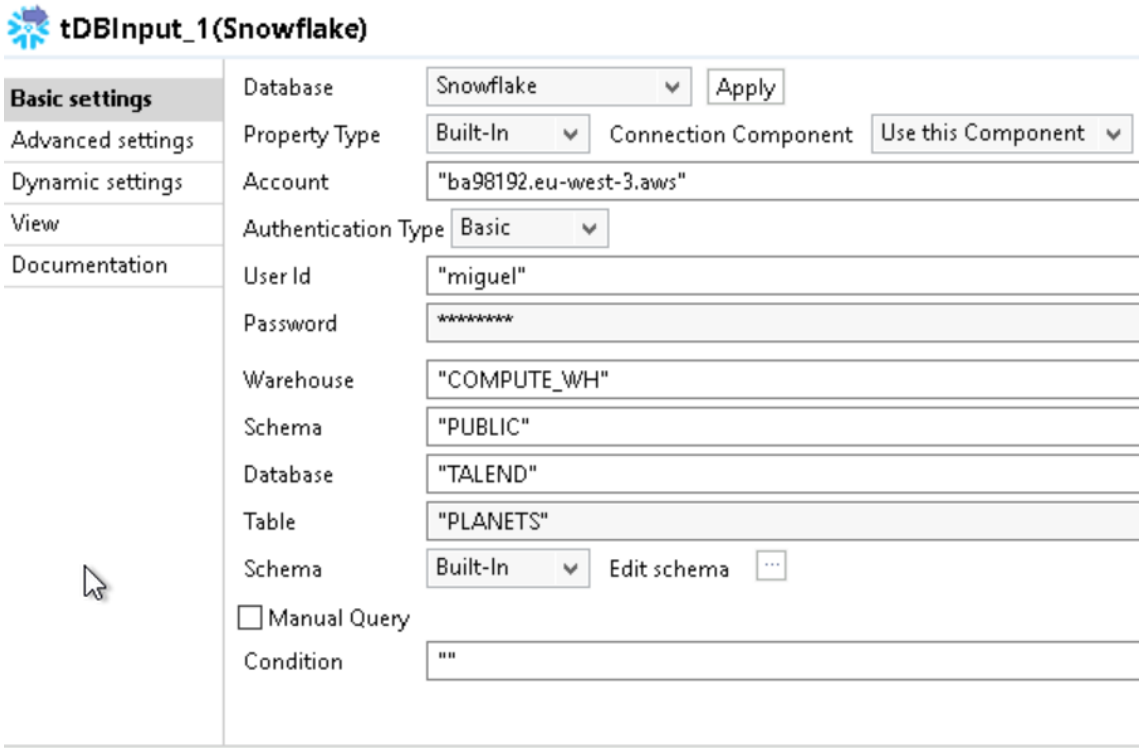
Un esquema de la arquitectura que sigue el job es el siguiente:

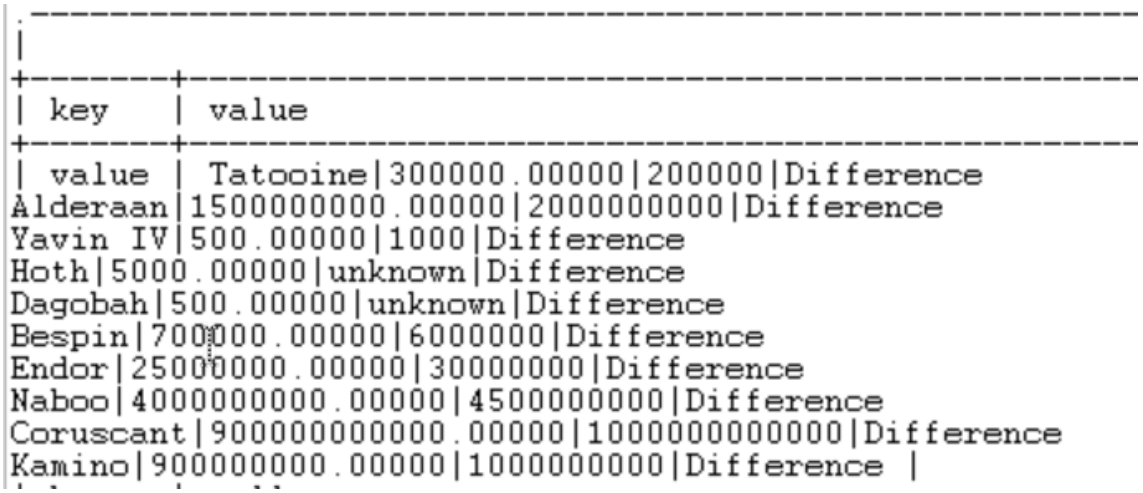


La configuración de los componentes esenciales de este trabajo se puede ver en las siguientes imágenes junto con la salida del mismo.




tExtractXMLField_1
Basic settings
Advanced settings
Dynamic settings
View
Documentation
Property Type
Built-In
Schema
Built-In
Edit schema
Sync columns
XML field
body
Loop XPath query
"/root/results"
Mapping
Column
XPath query
population
"population"
name
"name"
orbital_period
"orbital_period"
climate
"climate"
terrain
"terrain"
surface_water
"surface_water"
rotation_period
"rotation_period"
Limit
Die on error


tDBInput_1(Snowflake)
Basic settings
Advanced settings
Dynamic settings
View
Documentation
Database
Snowflake
Apply
Property Type
Built-In
Connection Component
Use this Component
Account
"ba98192.eu-west-3.aws"
Authentication Type
Basic
User Id
"miguel"
Password
****
Warehouse
"COMPUTE_WH"
Schema
"PUBLIC"
Database
"TALEND"
Table
"PLANETS"
Schema
Built-In
Edit schema
Manual Query
Condition
""



```
+-------+-------------------------------------------------
| key   | value
+-------+-------------------------------------------------
| value | Tatooine|300000.00000|200000|Difference
Alderaan|1500000000.00000|2000000000|Difference
Yavin IV|500.00000|1000|Difference
Hoth|5000.00000|unknown|Difference
Dagobah|500.00000|unknown|Difference
Bespin|700000.00000|6000000|Difference
Endor|2500000000.00000|30000000|Difference
Naboo|4000000000.00000|4500000000|Difference
Coruscant|900000000000.00000|1000000000000|Difference
Kamino|900000000.00000|1000000000|Difference |
```

Algunas de las ventajas de trabajar con UDFs en lugar de ejecutar el job de Talend por separado son las siguientes:

1) Centralización del proceso: En el caso de que la empresa use Snowflake como herramienta de almacenamiento y tratamiento de datos, todo quedaría centralizado en esta plataforma, con todas las ventajas y facilidades asociadas a ello.

2) Facilidad de acceso para usuarios: No hace falta tener conocimientos de Talend para ejecutar el proceso, ya que esto se puede hacer con una consulta SQL sencilla.

3) Portabilidad: El .jar es un artefacto portable que puede ser usado en otros sistemas compatibles con Java.

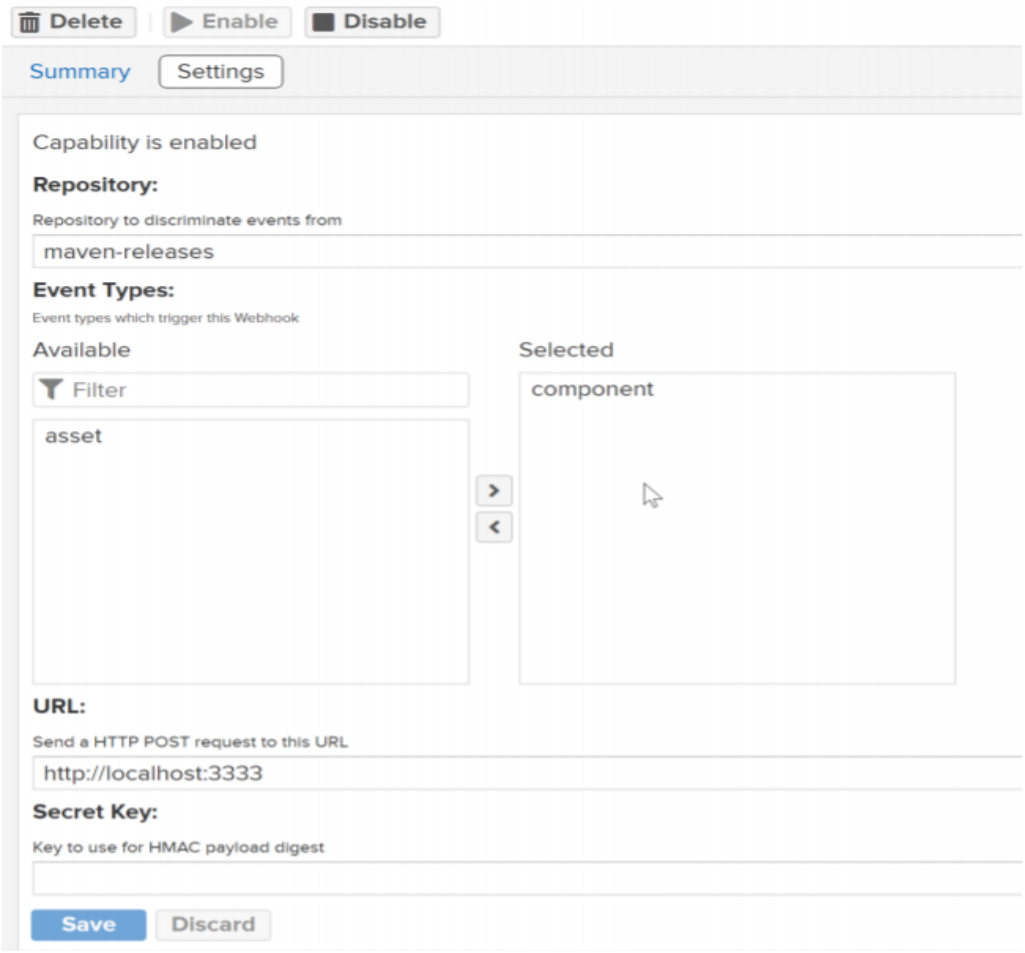
4) Menor dependencia de Talend al no ser necesario tenerlo instalado salvo para la configuración inicial del job. A partir de ese momento, Talend solo sería necesario para realizar modificaciones al job original.

5) Rendimiento y escalabilidad: Snowflake utiliza toda su infraestructura para optimizar la ejecución de las UDF.

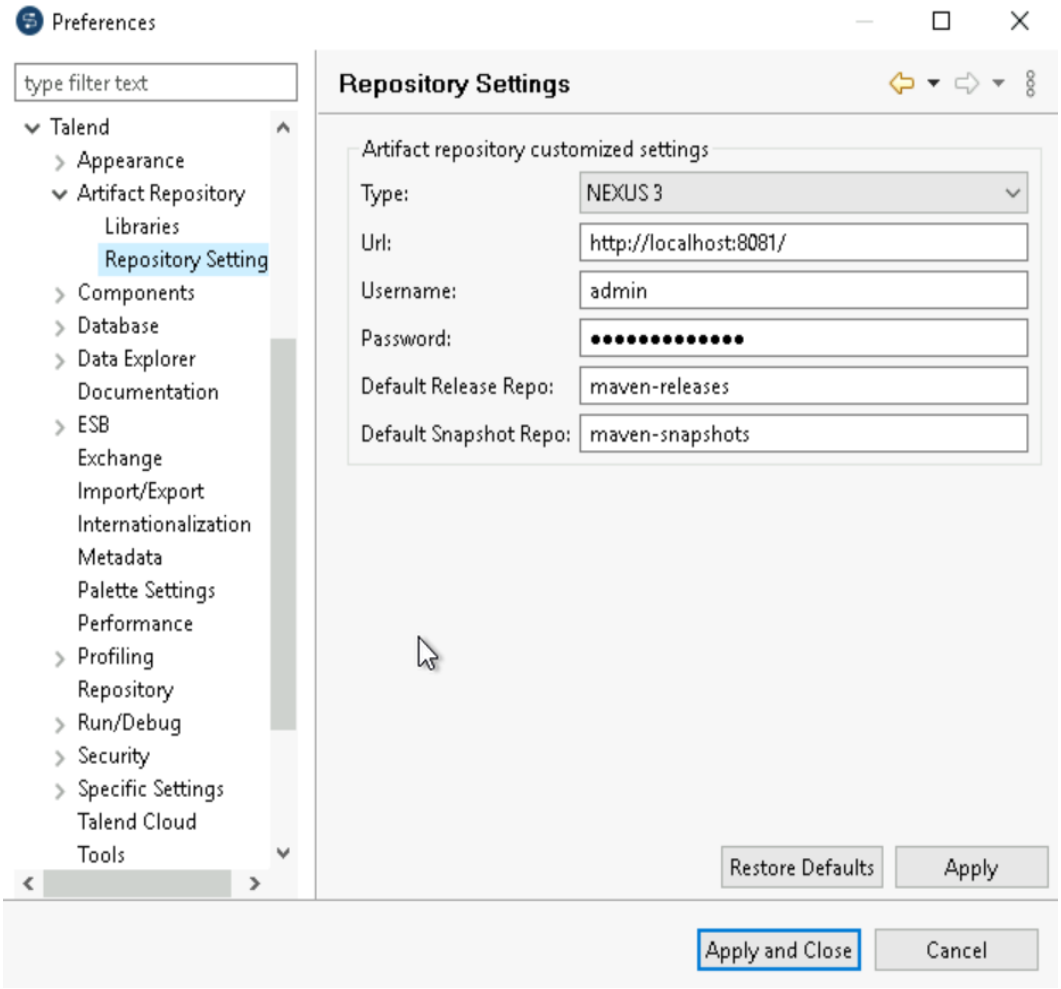
CONFIGURACIÓN DE NEXUS REPOSITORY

1) En nuestro caso (entorno de pruebas), descargar una instalación local del repositorio de nexus Download
2) Descomprimir el .zip de Nexus y ejecutarlo (en Windows con: nexus.exe /run)
3) Navegar a localhost:8081 e iniciar sesión. Para ello hay que introducir como username: admin y como password la que nos dan en el archivo \sonatype-work\nexus3\admin.password

4) Una vez dentro de nuestro repositorio, hemos de configurar un nuevo webhook. Para ello navegamos a settings/system/capabilities/create capability/webhook repository y lo configuramos como aparece en la siguiente imagen.



5) Añadir la configuración con el repositorio de Nexus a Talend. Para ello navegar a preferences/Talend/Artifact Repository/Repository Settings y una vez ahí configurar como se observa en la imagen.



DESCARGA Y CONFIGURACIÓN DE APACHE MAVEN

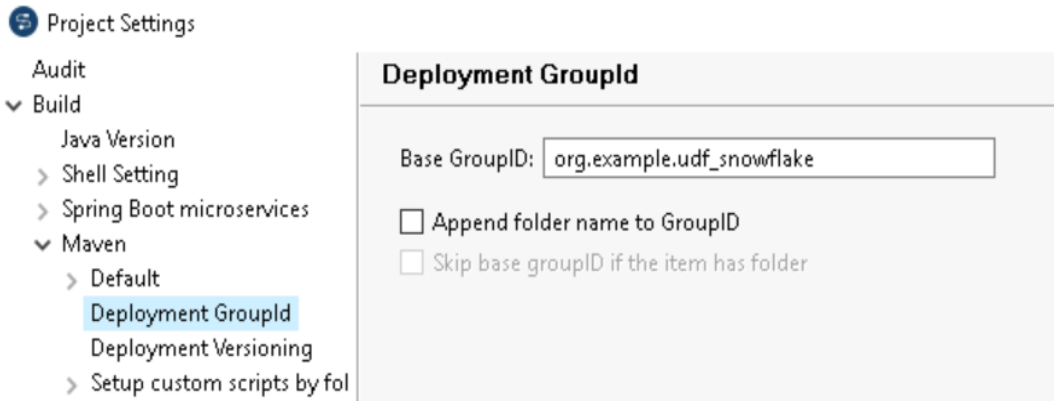
1) Descargar Apache Maven [Download Apache Maven – Maven]
2) Descomprimir la carpeta e incluirla en el path del sistema
3) Modificar el archivo de configuración de maven settings.xml ubicado en la carpeta /conf incluyendo lo siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>

<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"

  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
http://maven.apache.org/xsd/settings-1.0.0.xsd">

  <servers>

    <!-- credentials to access the default Nexus releases/snapshots -->
```

```
    <!-- to be used in -DaltdeploymentRepository <id> parameter -->
    <server>
      <id>maven-releases</id>
      <username>admin</username>
      <password>XXXXX</password>
    </server>
    <server>
      <id>maven-snapshots</id>
      <username>admin</username>
      <password>XXXXX</password>
    </server>
  </servers>
<profiles>
  <profile>
    <id>talend-ci</id>
    <!-- Nexus Settings / Credentials -->
    <properties>
      <nexus_host>http://localhost:8081</nexus_host>
    </properties>
    <!-- Nexus Settings / Credentials -->
    <repositories>
      <!-- For Nexus / JFrog -->
      <repository>
        <id>maven-releases</id>
        <name>maven-releases</name>
        <url>${nexus_host}/repository/maven-releases/</url>
```

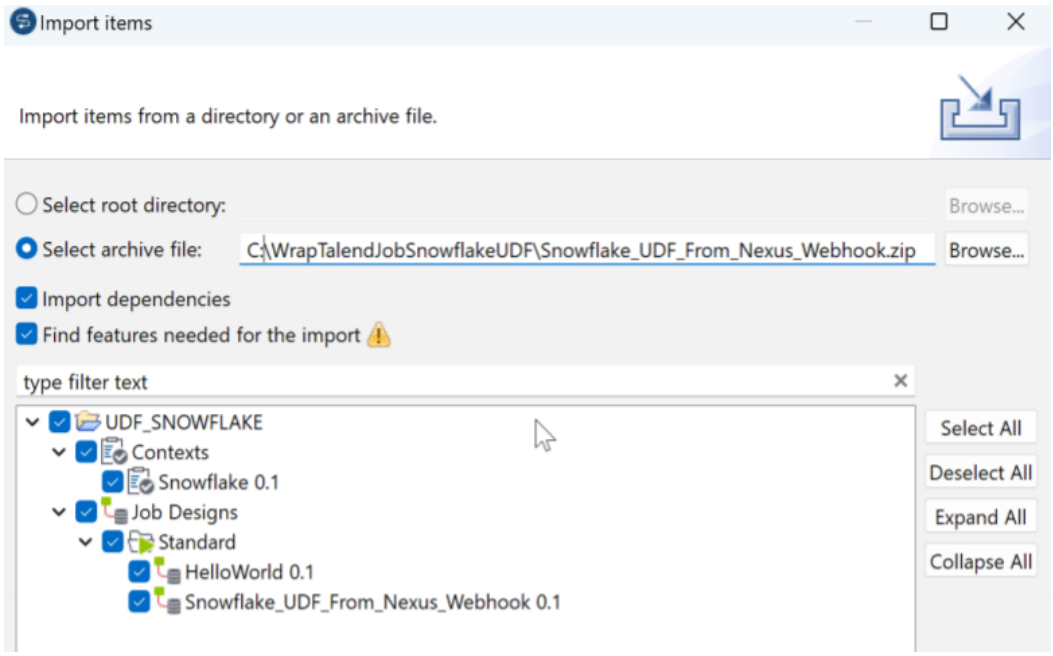
```
        <layout>default</layout>
      </repository>
    </repositories>
  </profile>
</profiles>
```

4) Ajustar la configuración de Maven en Talend como en la imagen: Para ello navegar al File/edit project properties/build/Maven y una vez ahí introducir como "Base GroupId": com.talend.udf_snowflake (esta variable debe coincidir con el groupId especificado en el pom.xml del proyecto)



CONFIGURACIÓN DEL PROYECTO DE MAVEN

1) Lo primero es crear un nuevo proyecto local llamado UDF-SNOWFLAKE. Una vez hecho esto importamos los jobs previamente descargados tal y como se observa en la siguiente imagen.



2) Configuramos las variables de contexto con las que se correspondan con nuestro repositorio de Nexus y nuestra cuenta de Snowflake.

	Name	Type	Comment	Default	
				Value	Enable prompt
1	git_working_folder	Directory		C:/temp/template-proje	☐
2	nexus_artifact	String		HelloWorld	☐
3	nexus_version	String		0.1.0	☐
4	nexus_group	String		talend	☐
5	Snowflake (from repository co				
6	snowflake_account	String		ba98192.eu-west-3.aws	☐
7	snowflake_database	String		TALEND	☐
8	snowflake_password	Password		**********	☐
9	snowflake_role	String			☐
10	snowflake_schema	String		PUBLIC	☐

	Name	Type	Comment	Default	
				Value	Enable prompt
1	git_working_folder	Directory		C:/temp/template-proje	☐
2	nexus_artifact	String		HelloWorld	☐
3	nexus_version	String		0.1.0	☐
4	nexus_group	String		talend	☐
5	Snowflake (from repository co				
6	snowflake_account	String		ba98192.eu-west-3.aws	☐
7	snowflake_database	String		TALEND	☐
8	snowflake_password	Password		**********	☐
9	snowflake_role	String			☐
10	snowflake_schema	String		PUBLIC	☐

Una vez hemos hecho todo esto, estamos preparados para instalar las dependencias necesarias para el proyecto.

DESCARGA DE LAS DEPENDENCIAS DE MAVEN

Una vez hemos descargado y configurado Maven, toca instalar las dependencias de este proyecto. Para ello, nos movemos a la carpeta descomprimida anteriormente (maven-dependency-plugin), navegamos a local-deps-plugin y, una vez ahí, ejecutamos el comando **mvn clean install**. Este comando se encarga de instalar todas las dependencias necesarias para poder ejecutar el proyecto correctamente.

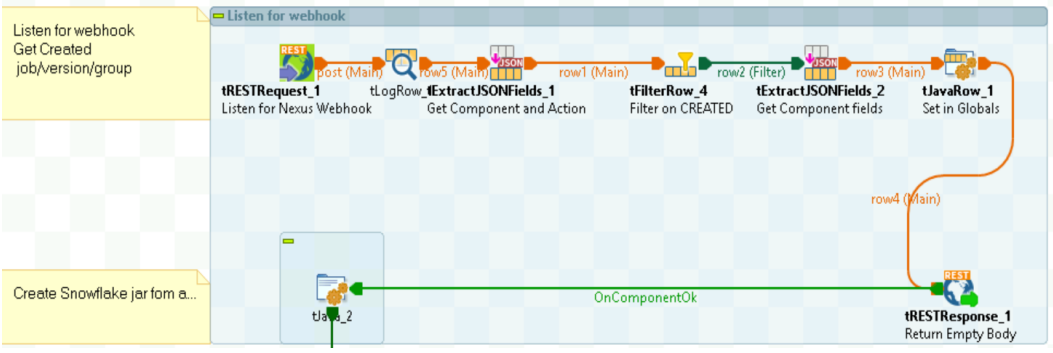
PUESTA EN MARCHA DEL PROYECTO

Para probar el correcto funcionamiento del proyecto:

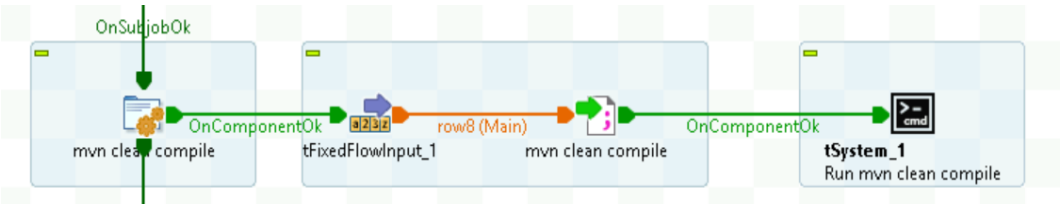
1) Lo primero es ejecutar el job Snowflake_UDF_From_Nexus_Webhook. Comprobamos que queda esperando a la publicación de un artefacto en Nexus.
2) En segundo lugar, hemos de publicar el artefacto, para ello hacemos clic derecho sobre el job que queremos publicar – publish y aceptamos.
3) La publicación del artefacto es detectada por el job orquestador y comienza la ejecución

   El Job realiza las siguientes operaciones:

   - Extrae la información del artefacto desde la solicitud.

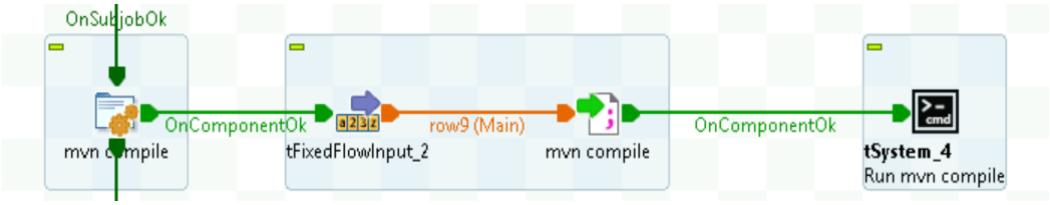


   - Ejecuta un comando de Maven para descargar las dependencias y generar la clase envuelta

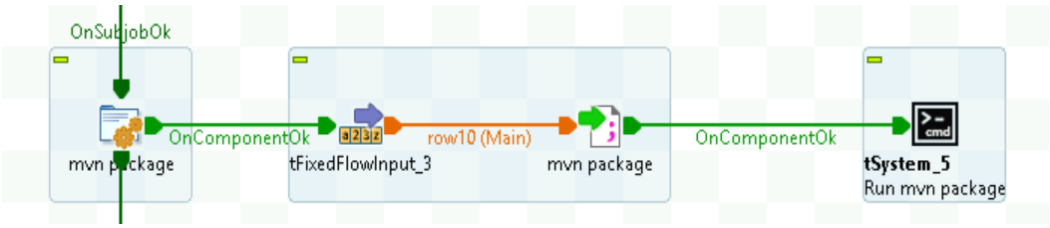


Durante este paso, pueden aparecer varios mensajes de error de compilación porque Maven necesita modificar el archivo pom.xml para actualizar las referencias de ${job.class}.

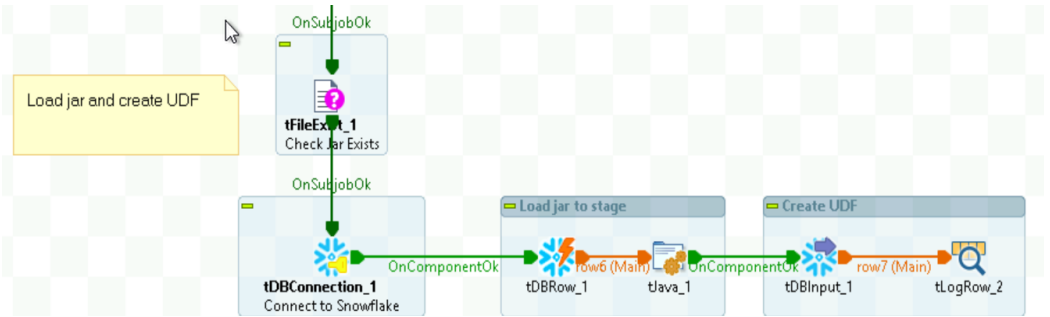
- Ejecuta un segundo comando de Maven para compilar la clase envuelta.



- Ejecuta un comando final de Maven para empaquetar la clase envuelta junto con sus dependencias en un archivo JAR.

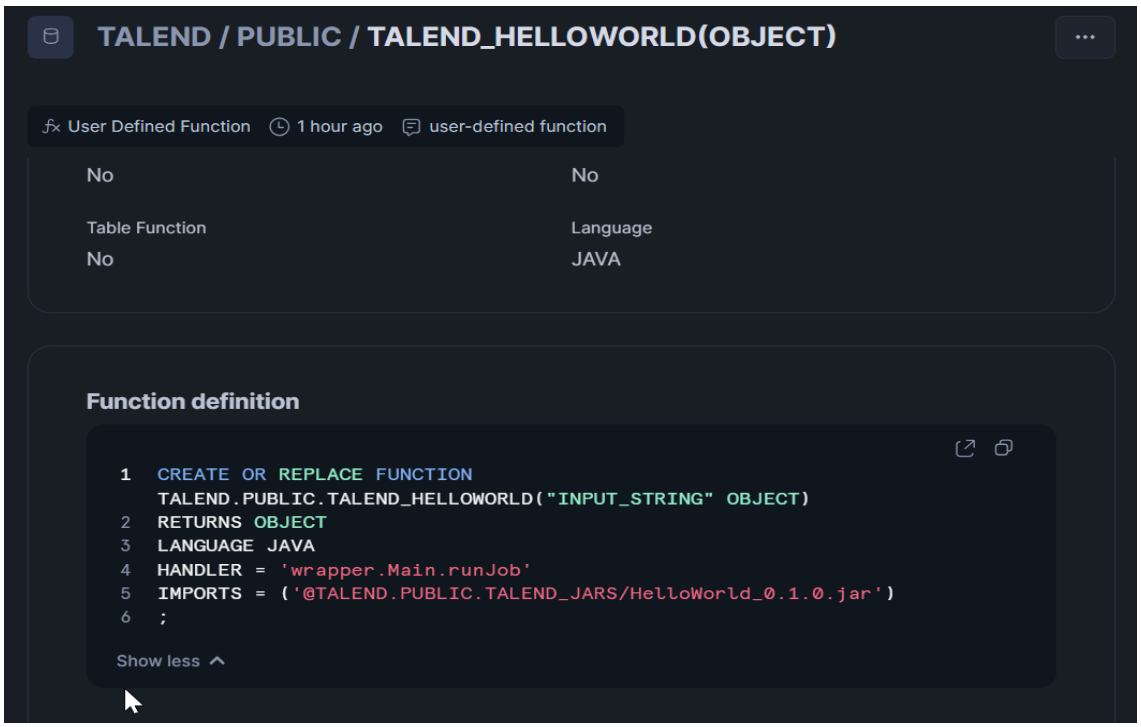


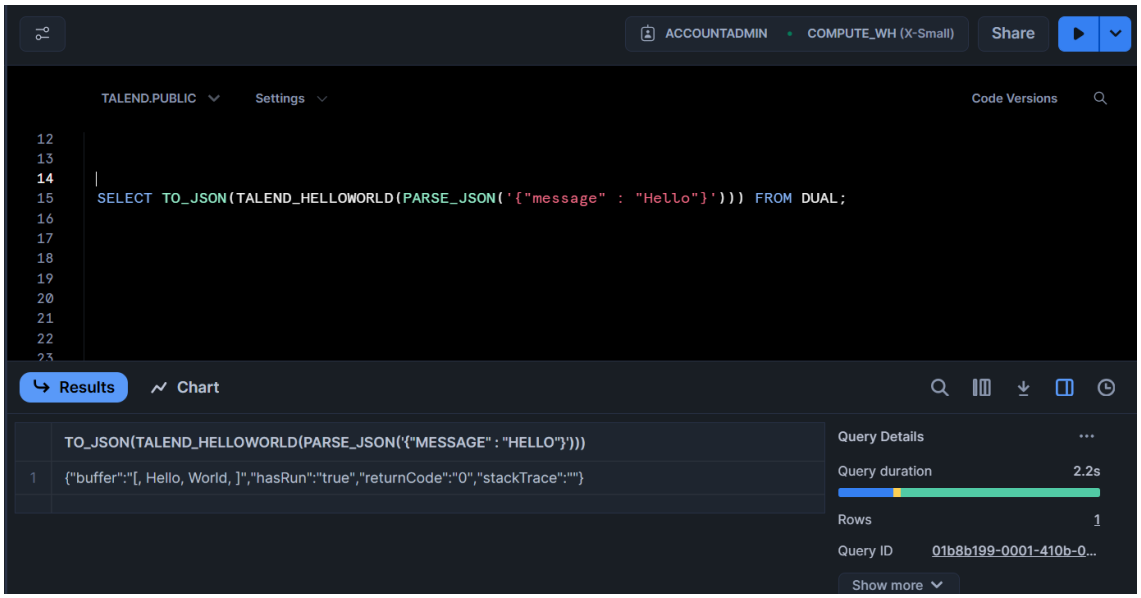
- Sube el archivo JAR a Snowflake y lo registra como una UDF.



En caso de que surjan errores durante la ejecución del job, conviene revisarlos, ya que normalmente son provocados por inconsistencias entre los nombres de las variables en el contexto del job de Talend y los nombres asignados en los Poms o en el repo de Nexus.

Finalmente, realizamos las últimas comprobaciones para verificar que la UDF se ha subido correctamente a Snowflake y se puede ejecutar.




ACCOUNTADMIN
COMPUTE_WH (X-Small)
Share
TALEND.PUBLIC
Settings
Code Versions
SELECT TO_JSON(TALEND_HELLOWORLD(PARSE_JSON('{"message" : "Hello"}'))) FROM DUAL;
Results
Chart
TO_JSON(TALEND_HELLOWORLD(PARSE_JSON('{"MESSAGE" : "HELLO"}')))
{"buffer":"[, Hello, World,]","hasRun":"true","returnCode":"0","stackTrace":""}
Query Details
Query duration
2.2s
Rows
1
Query ID
01b8b199-0001-410b-0...
Show more

BIBLIOGRAFÍA

Talend. (2023). *Wrapping a Talend Job as a Snowflake UDF Function*. Talend Academy. https://academy.talend.com/student/activity/1733236-wrapping-a-talend-job-as-a-snowflake-udf-function?sid=a0945e31-fc46-4a59-8c54-0ccb73f0b314&sid_i=2#/page/65830d2616980ac48bd6e47f

Snowflake Inc. (n.d.). *User-defined functions (UDFs) overview*. Documentación de Snowflake. https://docs.snowflake.com/en/developerguide/udf/udfoverview

Snowflake Inc. (n.d.). *Stored procedures vs. user-defined functions (UDFs)*. Documentación de Snowflake. https://docs.snowflake.com/en/developerguide/stored-procedures-vs-udfs

Qlik. (n.d.). *Boundless capabilities with Snowflake UDF, Snowpark, and Talend*. Qlik Customer Portal. https://customerportal.qlik.com/article/Boundless-capabilities-with-Snowflake-UDF-Snowpark-and-Talend