Artículo
Ricardo Paiva · 24 jun, 2021 Lectura de 6 min

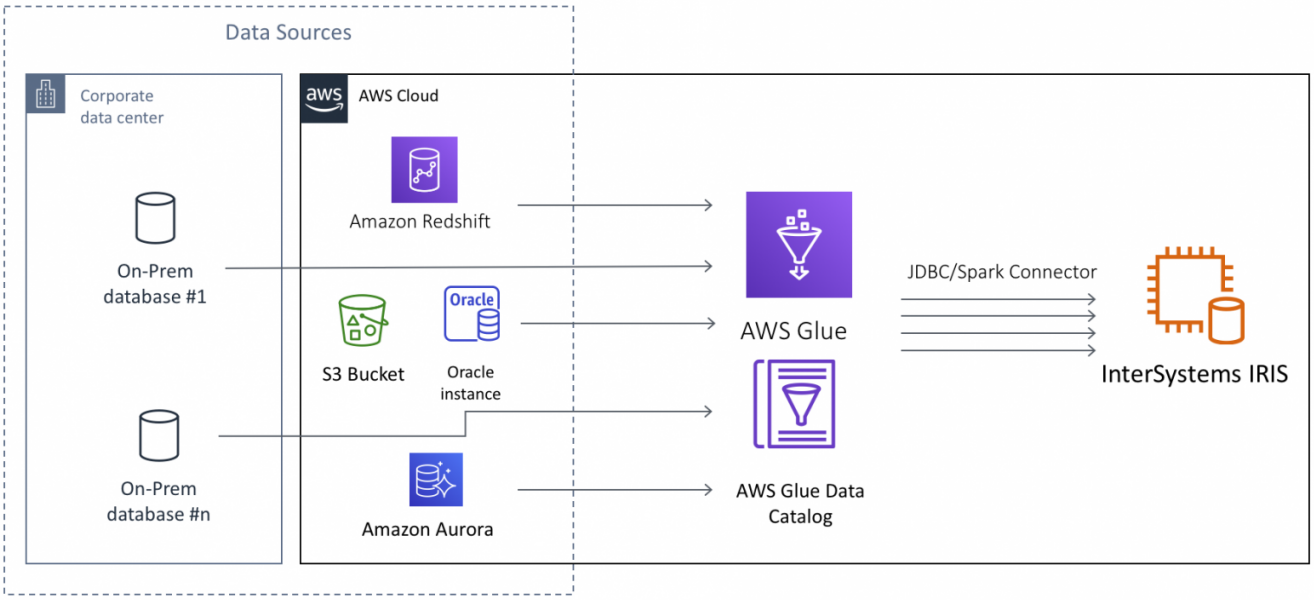
Cómo utilizar AWS Glue con InterSystems IRIS

Publicación Original por: Anton Umnikov

Arquitecto Senior de soluciones en la nube en InterSystems
AWS CSAA, GCP CACE

AWS Glue es un proceso ETL (extraer, transformar y cargar) completamente gestionado, que hace sencillo y rentable clasificar los datos, limpiarlos, enriquecerlos y moverlos de forma fiable entre diferentes almacenes de datos.

En el caso de InterSystems IRIS, AWS Glue permite mover grandes cantidades de datos a IRIS desde fuentes de datos tanto en la nube como en las propias instalaciones (*on-premise*). Las fuentes de datos potenciales incluyen, pero no se limitan a, bases de datos *on-prem*, archivos CSV, JSON, Parquet y Avro que residen en buckets S3, bases de datos nativas en la nube como AWS Redshift y Aurora, y muchas otras.



Este artículo asume que tienes un conocimiento básico de AWS Glue, al menos en el nivel para completar los Tutoriales de introducción a AWS Glue . Nos centraremos en los aspectos técnicos de la configuración de los trabajos (*Jobs*) en AWS Glue para utilizar InterSystems IRIS como Datos Objetivo, o en otros términos, "receptor de datos".

Imagen de https://docs.aws.amazon.com/glue/latest/dg/components-key-concepts.html

Los Trabajos en AWS Glue se ejecutan "sin servidor". Todos los recursos necesarios para realizar el trabajo son suministrados de forma dinámica por AWS solo durante el tiempo en que el trabajo se ejecuta realmente; y se destruyen en el momento en que el trabajo se completa. Por eso, en lugar de provisionar, gestionar e incurrir en costes continuos por la infraestructura necesaria, se te factura solo por el tiempo en que el trabajo realmente se ejecuta y puedes dedicar tus esfuerzos a escribir el código del Trabajo. Durante el tiempo de "inactividad" no se consumen recursos, salvo los buckets S3, que almacenan el código del Trabajo y la configuración.

Aunque hay varias opciones disponibles, normalmente los Trabajos en Glue se ejecutan con un suministro dinámico en Apache Spark y se escriben en código PySpark. El Trabajo en Glue contiene la sección *"Extraer"* (donde los datos se extraen de las fuentes de datos), una serie de *"Transformaciones"* (construidas utilizando la API de Glue) y finalmente la parte *"Cargar"* o *"receptor"*, en la que después de la transformación final los datos se escriben en el sistema objetivo.

Para permitir que AWS Glue interactúe con IRIS necesitamos asegurar lo siguiente:

- Glue tiene acceso a la red de las instancias IRIS involucradas
- El archivo JAR del controlador JDBC de IRIS es accesible para el Trabajo en Glue
- El Trabajo en Glue utiliza la API compatible con InterSystems IRIS JDBC

Examinemos cada uno de los pasos necesarios.

Crear una conexión a IRIS

En la consola AWS, selecciona AWS Glue->Connections->Add Connection

Introduce el nombre de tu conexión y selecciona "JDBC" como Tipo de Conexión.

En la URL de JDBC introduce la cadena de conexión JDBC para tu instancia de IRIS, el nombre de usuario y la contraseña.

El siguiente paso es crucial, ***necesitas asegurarte de que Glue coloca sus endpoints en el mismo VPC que tu instancia de IRIS.*** Selecciona la VPC y la Subred de tu instancia de IRIS. Cualquier grupo de seguridad con una regla de entrada de autorreferencias para todos los puertos TCP se ejecutaría aquí. Por ejemplo, el grupo de seguridad de tu instancia de IRIS.

Rol de IAM con acceso al controlador JDBC

Si aún no lo has hecho, carga el archivo JAR del controlador JDBC de IRIS intersystems-jdbc-3.0.0.jar en el bucket S3. En este ejemplo, estoy usando el bucket s3://irisdistr. Sería distinto para tu cuenta.

Necesitas ***crear un rol de IAM para tu Trabajo en Glue, que pueda acceder a ese archivo,*** junto con otros buckets S3 que Glue utilizaría para almacenar scripts, registros, etc.

Asegúrate de que tiene acceso de lectura al bucket de tu controlador JDBC. En este caso, concedemos a esta función (GlueJobRole) acceso de solo lectura (ReadOnly) a todos los buckets, junto con el AWSGlueServiceRole predefinido. Puedes elegir limitar aún más los permisos de este rol.

Crear y configurar el Trabajo en Glue

Crear un nuevo Trabajo en Glue. Selecciona el rol de IAM, que creamos en el paso anterior. Deja todo lo demás de forma predeterminada.

En "Security configuration, script libraries, y job parameters (optional)" establece en "Dependent jars path" la ubicación del intersystems-jdbc-3.0.0.jar en el bucket S3.

Para la fuente: utiliza una de tus fuentes de datos preexistentes. Si seguiste los tutoriales mencionados más arriba, ya tendrás al menos una.

Utiliza la opción "Create tables in your data target" y selecciona la conexión IRIS que has creado en el paso anterior. Deja todo lo demás de forma predeterminada.

Si nos has seguido hasta ahora, deberías llegar a una pantalla similar a esta:

¡Ya queda poco! Solo necesitamos hacer un sencillo cambio en el *script* para cargar datos en IRIS.

Cómo ajustar el *script*

El *script* que Glue generó para nosotros utiliza AWS Glue Dynamic Frame, extensión propietaria de AWS para Spark. Aunque ofrece algunos beneficios para los trabajos de ETL, también garantiza que no puedes escribir datos en ninguna base de datos para la que AWS no haya administrado la oferta de servicios.

Buenas noticias- en el momento de escribir los datos en la base de datos ya no son necesarios todos los beneficios de Dynamic Dataframe, como no aplicación de esquema para los datos "sucios" (en el momento de escribir los datos se supone que están "limpios") y podemos convertir fácilmente Dynamic Dataframe a un Dataframe nativo en Spark que no está limitado a los objetivos administrados por AWS y puede trabajar con IRIS.

Así que la línea que necesitamos cambiar es la línea 40 de la imagen más arriba. Un último consejo.

Este es el cambio que necesitamos hacer:

```
#datasink4 = glueContext.write_dynamic_frame.from_jdbc_conf(frame = dropnullfields3,
catalog_connection = "IRIS1", connection_options = {"dbtable": "elb_logs", "database"
: "USER"}, transformation_ctx = "datasink4")
dropnullfields3.toDF().write \
    .format("jdbc") \
    .option("url", "jdbc:IRIS://172.30.0.196:51773/USER/") \
    .option("dbtable", "orders") \
    .option("user", irisUsername) \
    .option("password", irisPassword) \
    .option("isolationlevel","NONE") \
    .save()
```

Donde irisUsername e irisPassword son el nombre de usuario y las contraseñas para tu conexión JDBC en IRIS.

Nota: *almacenar las contraseñas en el código fuente NO es conveniente. Te animamos a utilizar herramientas como AWS Secrets Manager para ello, pero entrar en este nivel de detalles de seguridad está más allá del alcance de este artículo. Este es un buen artículo sobre el uso de AWS Secrets Manager en AWS Glue.*

Ahora pulsa el botón "Run Job", siéntate y relájate mientras AWS Glue efectúa el proceso ETL por ti.

Bueno... lo más probable es que cometas algunos errores al principio... Todos sabemos cómo funciona. Una errata por aquí, un puerto equivocado en el grupo de seguridad por allí... AWS Glue utiliza CloudWhatch para almacenar todos los registros de ejecución y de errores. Busca grupos de registro en: */aws-glue/jobs/error* y */aws-glue/jobs/output*, para identificar lo que salió mal.

¡Disfruta de la programación de los procesos ETL en la nube!

#AWS #Bases de datos #Big Data #Mejores prácticas #Nube #Python #SQL #InterSystems IRIS