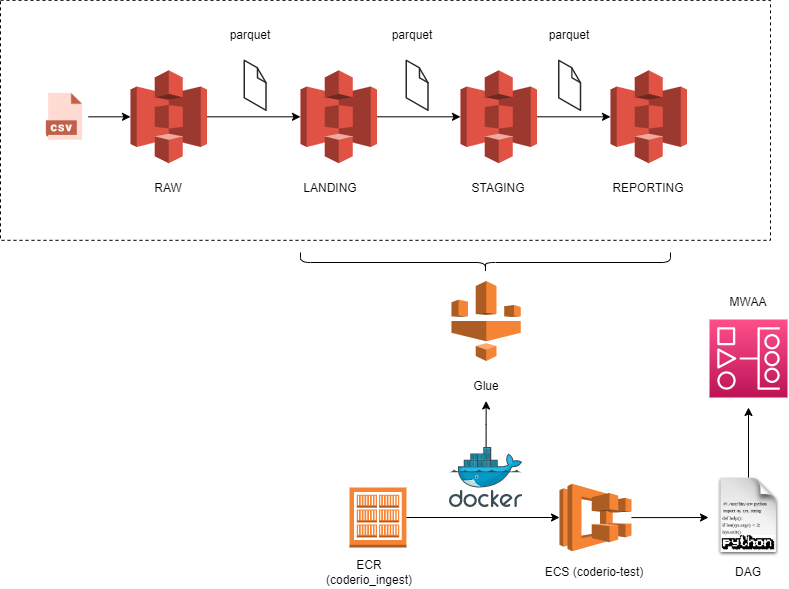
**Arquitectura propuesta:**

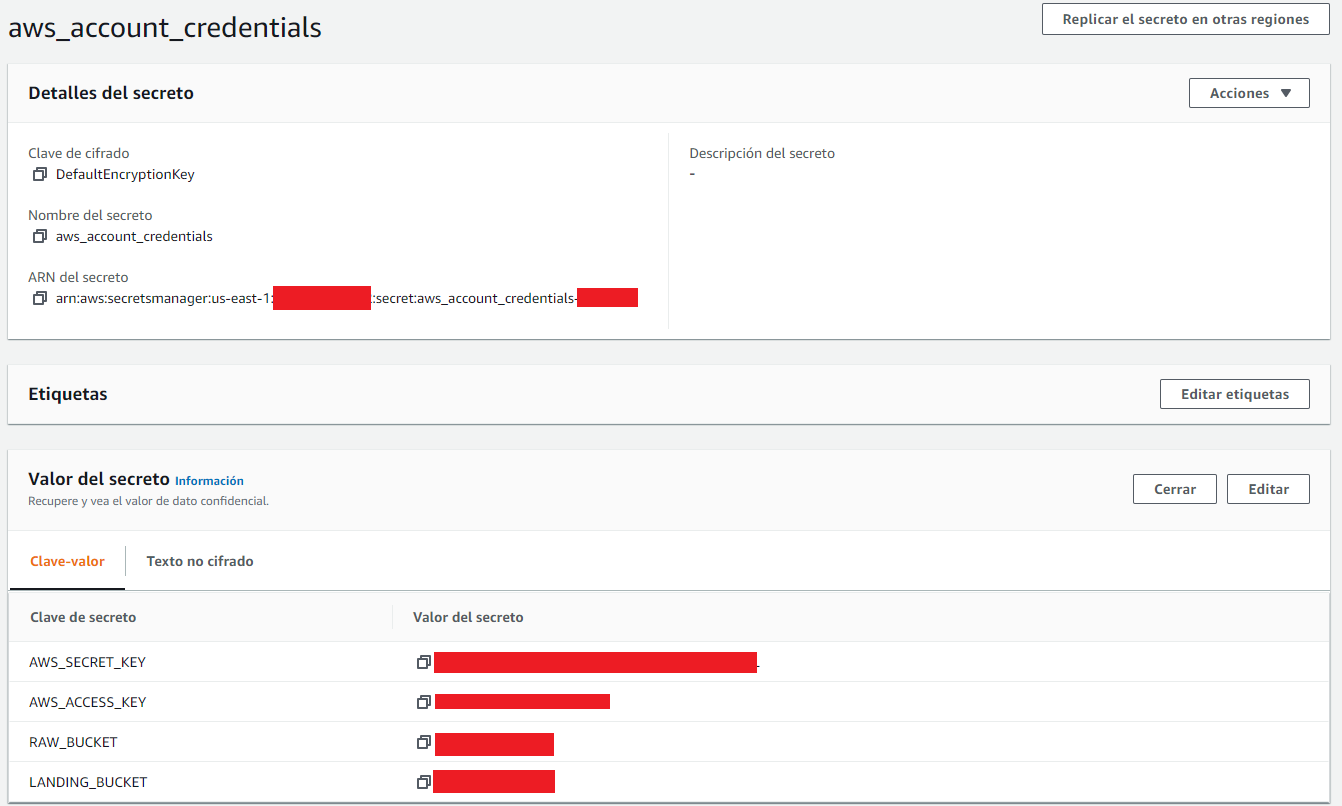


**Servicios AWS:**

***Secrets Manager:*** Elegir como tipo de secreto “Otro tipo de secreto”, definir los siguientes valores:

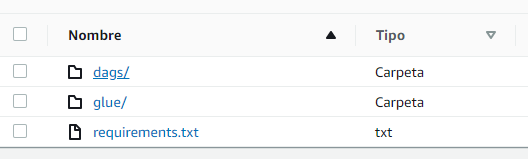
* AWS\_SECRET\_KEY
* AWS\_ACCESS\_KEY
* RAW\_BUCKET
* LANDING\_BUCKET

El nombre del secreto es **aws\_account\_credentials**.



***S3:*** Se tiene la siguiente composición de buckets:

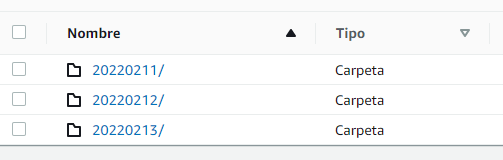
* **codeChallenge20220211**: en este bucket se tiene la carpeta **dags** usada para almacenar los scripts de **Airflow**, el archivo de requerimientos y una carpeta **glue** que se usa para almacenar el código de los Jobs de Glue creados para la solución.



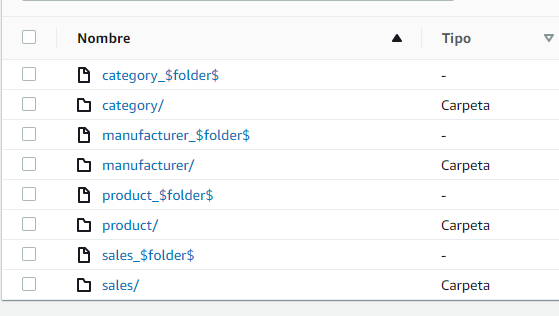
* **rawBucket20220211**: en este bucket se tienen dos carpetas: *historic* e *incremental* en las que se almacenará el input de data histórica e incremental para el proceso. Para este caso particular se asume que los archivos .zip incrementales son sobreescritos.

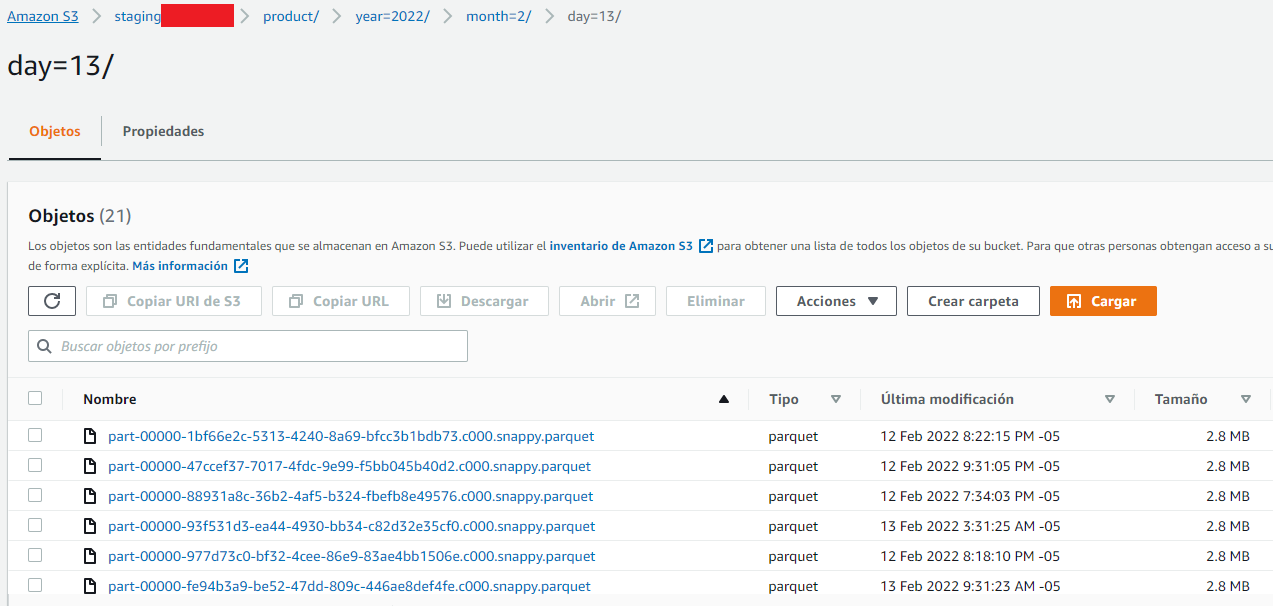


* **landing20220211**: en este bucket se crean como parte del proceso carpetas con el nombre YYYYMMDD en las que se almacena en formato parquet los archivos CSV del bucket raw. Para los archivos incrementales se agrega un sufijo **\_DELTA** al final del nombre.

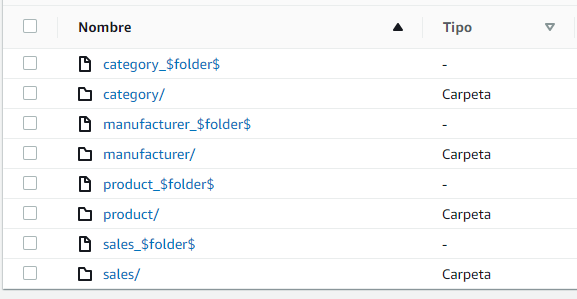


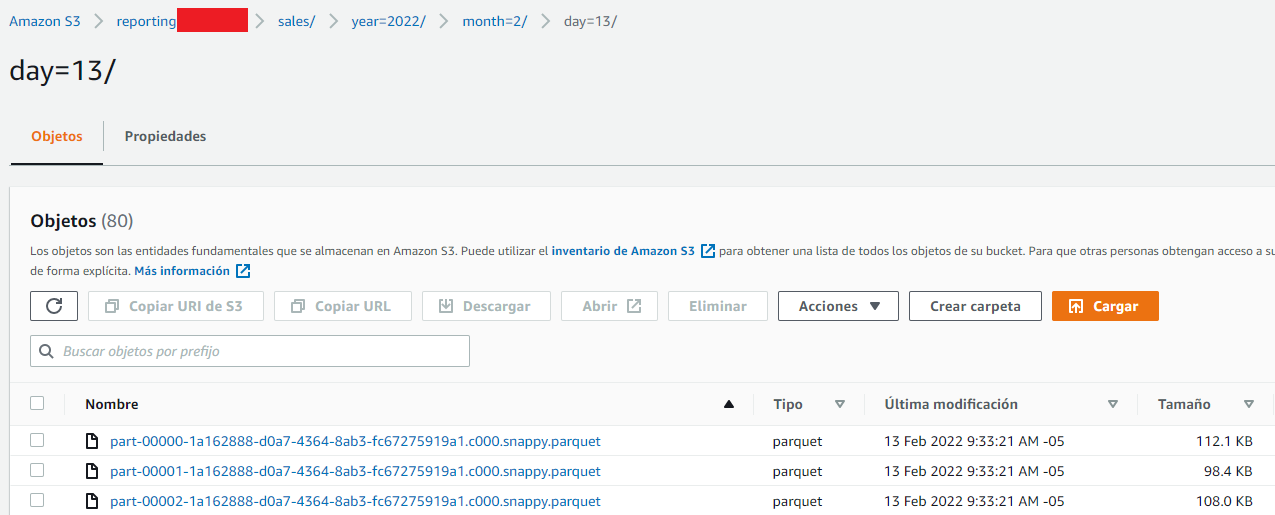
* **staging20220211**: en este bucket se tiene una carpeta que representa a cada tabla del modelo (particionada por year, month, day). Cada carpeta almacena la información completa (histórica e incremental) de las tablas en formato parquet.





* **reporting20220211**: este bucket contiene la información final del modelo (se aplica la lógica para eliminación de duplicados) particionada por year, month, day.

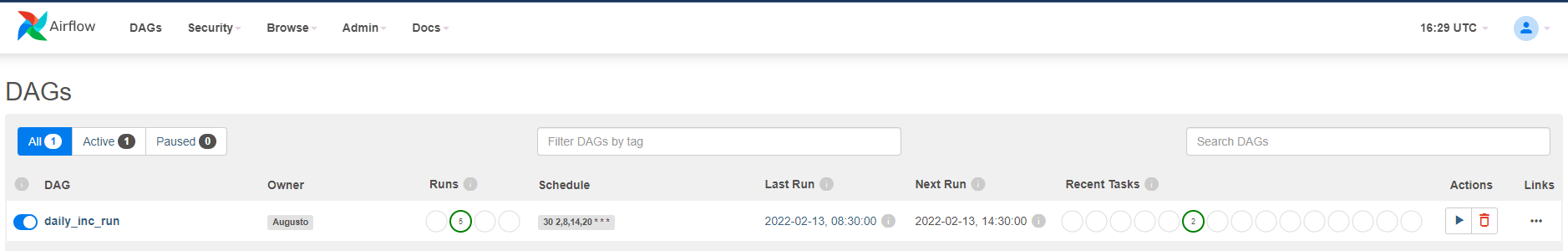




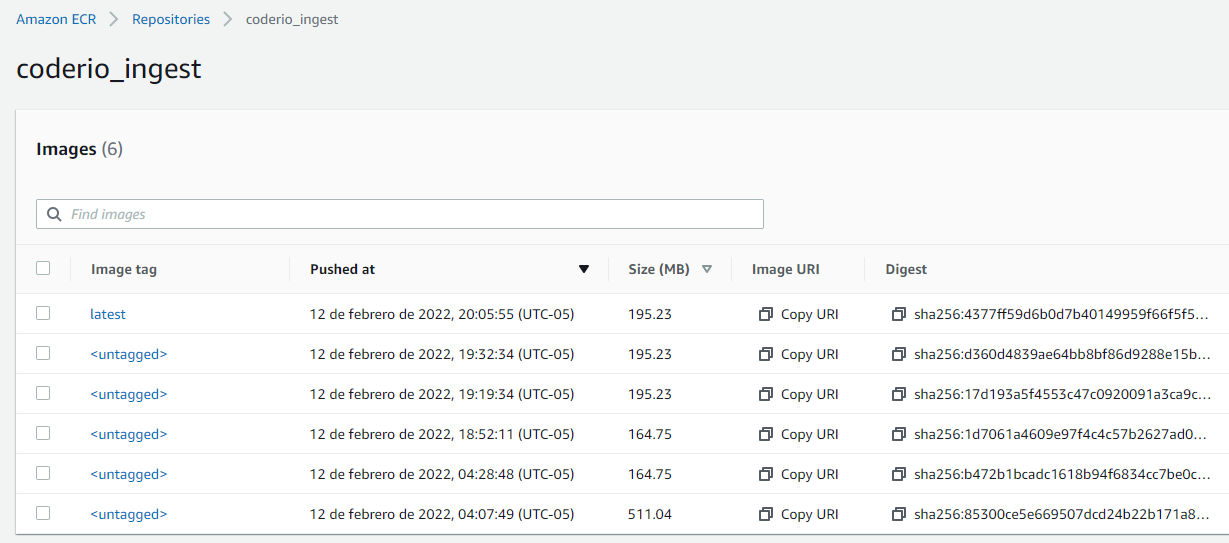
***MWAA:*** Usado para orquestar la ejecución del DAG que gatilla el ECS encargado de la carga histórica/incremental de la data para el modelo.

Para su creación, definir valores por defecto y crear VPC por default, crear el rol de ejecución por defecto y luego asignarle permisos para acceder a los losg de CloudWatch y ejecución de ECS. Definir el acceso del webserver como público.

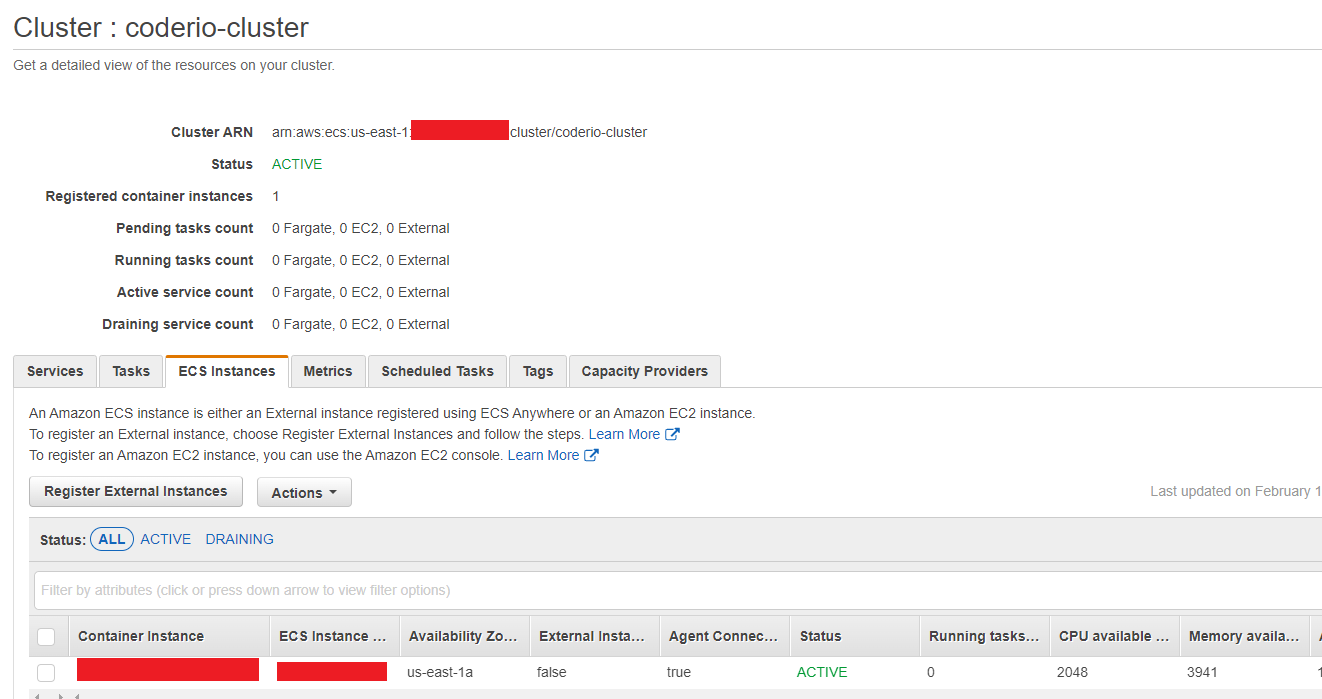
La ruta de la carpeta con los códigos de los dags en S3 es **codeChallenge20220211/dags/**



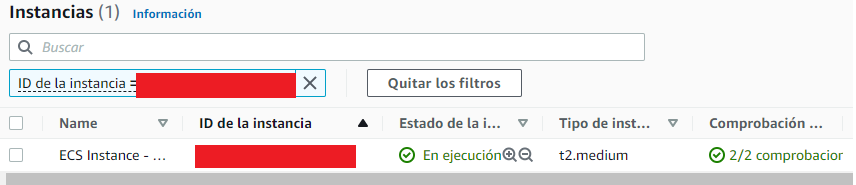
***ECR:*** Crear este servicio con los valores por defecto, el nombre del repositorio para nuestro caso es **coderio\_ingest**. Los comandos para el build y push de la imagen Docker nos lo brinda la interfaz de AWS.



***ECS:*** Crear un cluster con el nombre **coderio-cluster**, elegir como template EC2 Linux + Networking, tipo de instancia para nuestro caso t2.medium, el resto de valores dejarlos por defecto.

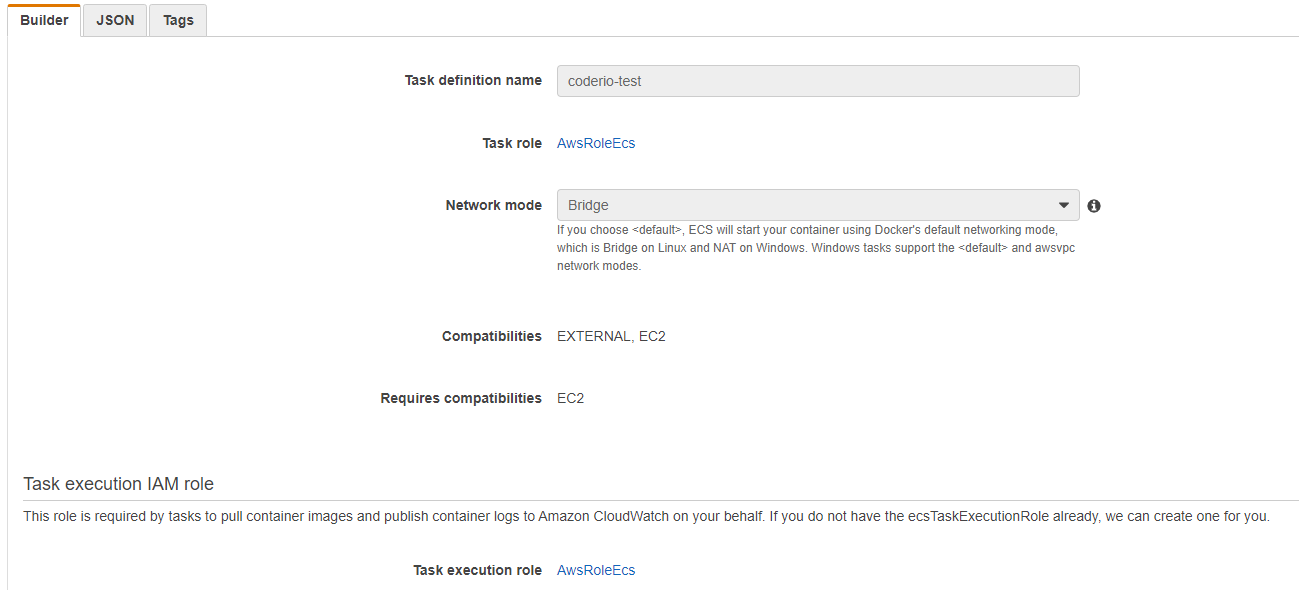
****

Para el grupo de seguridad de la instancia ECS configurar en reglas de entrada allow all traffic protocolo TCP con rango de puertos 0-65535 con origen 0.0.0.0/0 (esta parte de la configuración es para poder acceder a la ruta de los archivos de S3, puede realizar una configuración más segura y con mayores restricciones para una implementación en ambiente real).

****

**Creación de ECS task:**

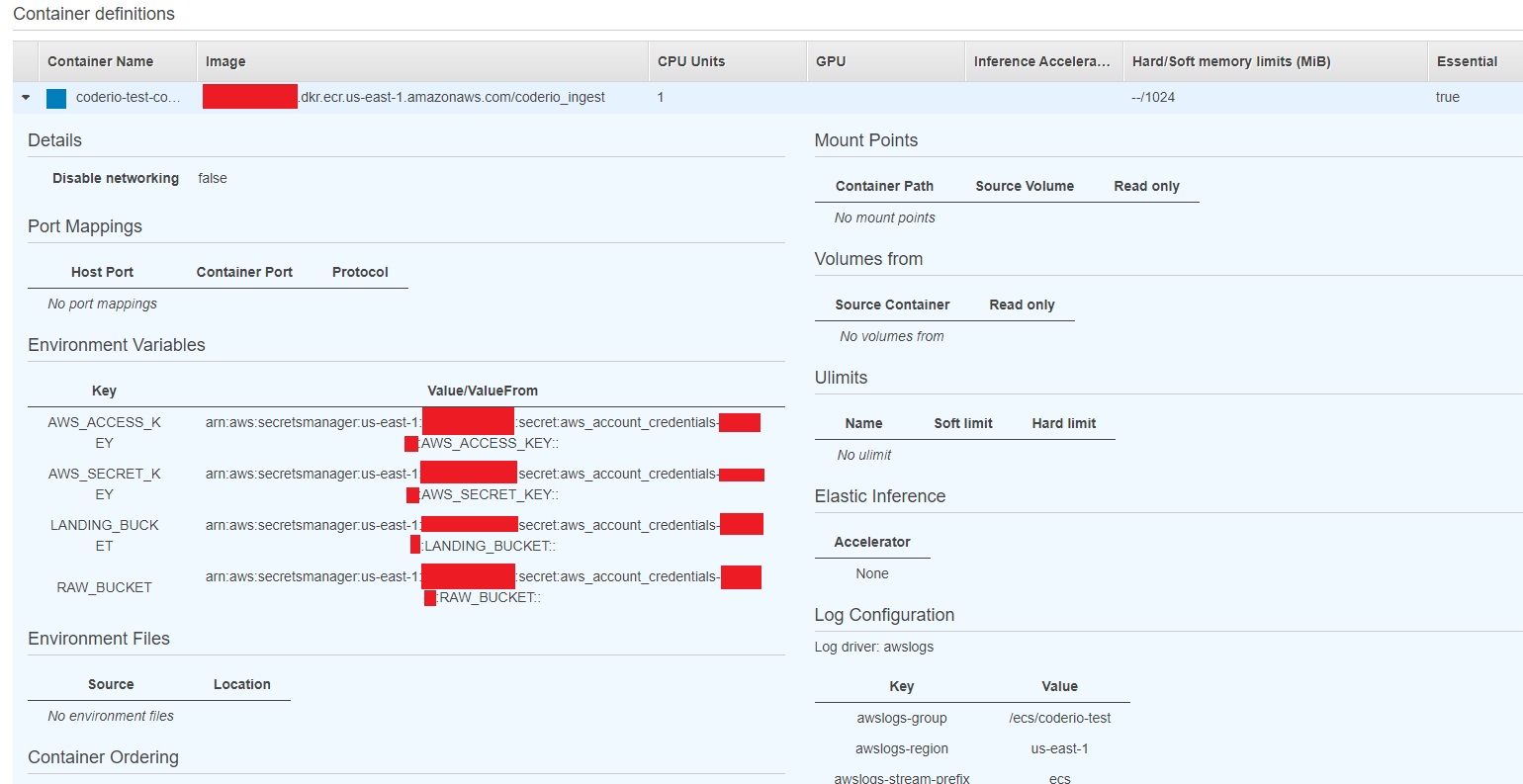
Crear un ECS task con el nombre **coderio-test**, el rol **AwsRoleEcs**, networking mode **Bridge**, compatibilidad **EC2**. Como task execution role definir también **AwsRoleEcs**.



**IAM Role:** AwsRoleEcs con políticas:

* SecretsManagerReadWrite
* AmazonS3FullAccess
* AmazonECSTaskExecutionRolePolicy
* AmazonEC2ContainerServiceforEC2Role

Para el caso de la definición del contenedor, usar el nombre **coderio-test-container**, asignarle la imagen de nuestro repositorio en ECR, definir como softmemory 1GB y variables de entorno las 4 definidas en el Secrets Manager.



***Glue:*** Se definieron 3 jobs:

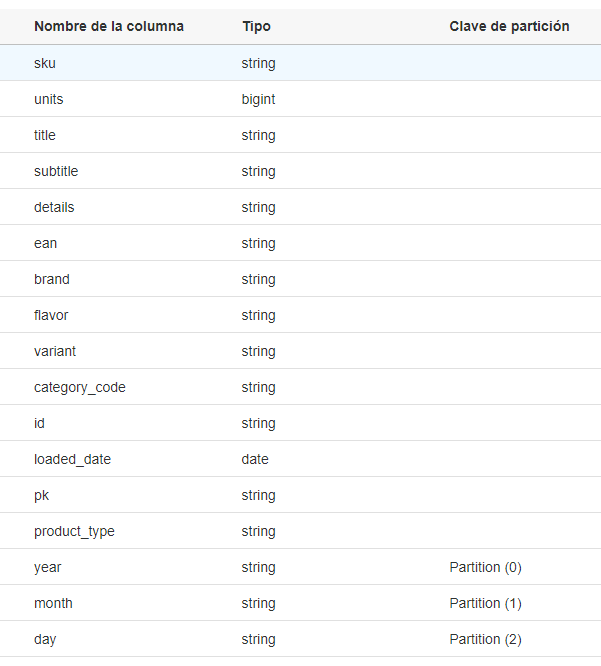
* historic\_load para la carga histórica.
* incremental\_load para carga incremental.
* remove\_duplicates para eliminación de duplicados y almacenado en el bucket de reporting.

En bases de datos se crearon dos:

* coderio\_db para información de staging.
* codeio\_db\_prod para información final.

En tablas se definieron 8 tablas (4 para cada ambiente), la estructura es la siguiente:

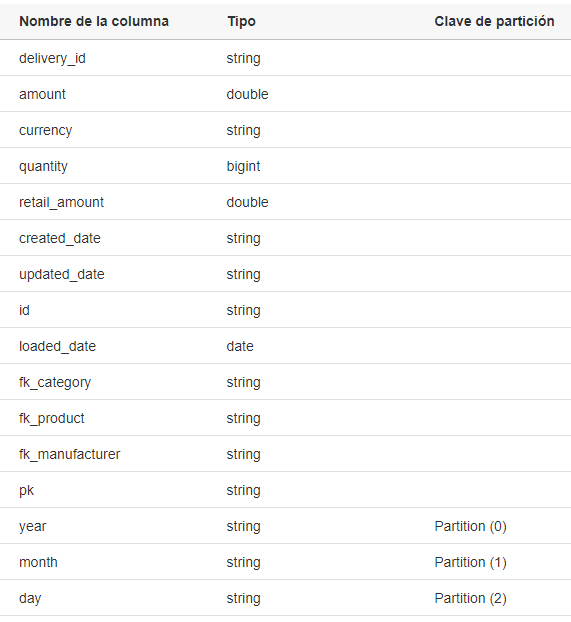
* **product** table:



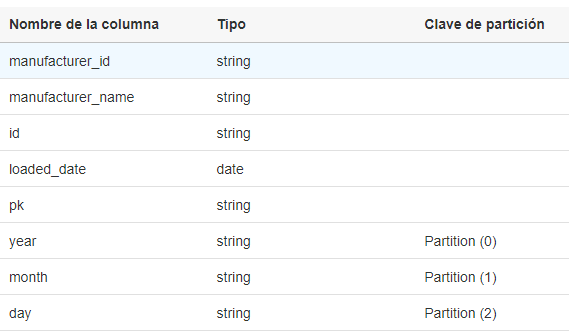
* **category** table:



* **sales** table:



* **manufacturer** table:



Adicionalmente se crearon dos crawlers, cada uno para actualizar/crear las tablas de staging/prod. Definir como table level 3 ya que la data se encuentra particionada a ese nivel (year/month/day/).

**Observaciones:**

* Se asume que la data incremental se carga 1 vez al día.
* Dentro de la data para las tablas no se pudo identificar un identificador único para registro, por lo que uso la combinación de la columna id de los inputs en combinación con otra (según la tabla) para garantizar la unicidad.

*Por ejemplo:*

En la tabla manufacturer para manufacturer id 0555EF06D716F9D5CA6385F21D3B5A0B se tienen dos manufacturer name 6FAA630E832B8774285E350FB518711C / E612211B9C50F99D0DE8758FFBCCA812.

De forma similar en la tabla de productos para el category code ARMA\_TU\_COMBO con category color ARG se tienen dos category name COMBOS EN OFERTA / ARMA TU COMBO