Georesearch 2022

Descripción breve

Documento guía para integrar las fases del proyecto

Proyecto CANIBALIZACIÓN

Modelo conceptual y automatización

Contenido

[Proceso de automatización 2022 3](#_Toc112782163)

[Bases de datos 3](#_Toc112782164)

[Tablas postgres 3](#_Toc112782165)

[Dns configurados: 3](#_Toc112782166)

[Endpoints 3](#_Toc112782167)

[Api keys 4](#_Toc112782168)

[Repositorio 4](#_Toc112782169)

[Api dms 5](#_Toc112782170)

[Preparación de datos previa migración 5](#_Toc112782171)

[Origen de los datos (Postgres) 5](#_Toc112782172)

[Destino de los datos 5](#_Toc112782173)

[Descubrimiento de nuevos datos 5](#_Toc112782174)

[Migración de tablas 5](#_Toc112782175)

[Solicitudes / Servicios 6](#_Toc112782176)

[Migración de tablas 6](#_Toc112782177)

[Creación de buffers 6](#_Toc112782178)

[Replicación de tablas 6](#_Toc112782179)

[Diagrama de arquitectura API DMS 7](#_Toc112782180)

[~~Migración con step Functions~~ Se incluye en el servicio replicate-tables 7](#_Toc112782181)

[Detalle de la solicitud: 8](#_Toc112782182)

[Diagrama de la arquitectura api estudios 9](#_Toc112782183)

[Lambda 10](#_Toc112782184)

[Endpoint 10](#_Toc112782185)

[Salida de resultados 10](#_Toc112782186)

[Solicitud 10](#_Toc112782187)

[~~Versión anterior~~ 10](#_Toc112782188)

[Ultima versión 11](#_Toc112782189)

[Lista de etls disponibles 12](#_Toc112782190)

[Diccionario de claves: 12](#_Toc112782191)

[customer\_list (requerido) [{},{}…{}] 12](#_Toc112782192)

[solicitud 12](#_Toc112782193)

[Subclave parámetros 13](#_Toc112782194)

[~~Futuros cambios~~ ya incluidos 13](#_Toc112782195)

[Flujo de actualización de datos 14](#_Toc112782196)

[Paso 1 Migración 14](#_Toc112782197)

[Paso 2 Creación de buffers 16](#_Toc112782198)

[Paso 3 Replicación 17](#_Toc112782199)

[Postman 18](#_Toc112782200)

[Agregar API KEY 19](#_Toc112782201)

[Actualizaciones de de la api para inputs a medida 19](#_Toc112782202)

# Proceso de automatización 2022

El objetivo de este proceso fue mejorar el actual proceso de generación de reportes de canibalización de la plataforma máster geo de manera que sea un proceso escalable y mantenible en el tiempo.

En este documento se cubren las etapas estándares de un ETL por medio de la generación de tablas en el Datawarehouse, extracción y procesamiento de datos, desarrollo de etls, desarrollo de apis y despliegue de información.

# Modelo conceptual

El origen del reporte de canibalización nace en la plataforma Mastergeo, desde ahí su objetivo es responder a la pregunta: ¿en cuánto afectan las ventas de una tienda o tiendas comerciales con respecto a la colocación de un nuevo local comercial en alguna zona geográfica?

De este análisis se pueden diferenciar 2 cosas, los locales comerciales y deliveries, ambos análisis requieren un precálculo que es entregado por un experto que tome la solicitud, junto a esto se solicitaran otros datos como el país, un área de búsqueda (buffer) , georreferencia del local a evaluar ,distancia, superficie mt2 , lista de categorías y subcadenas a buscar.

# Bases de datos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Motor** | **Ambiente** | **Ip** | **Usuario** | **db** |
| PostgreSQL | PROD | 10.10.0.40 | postgres | mastergeo\_countries |
| PostgreSQL | QA | 10.10.0.52 | postgres | mastergeo\_countries |
| Athena | QA |  |  | qa\_countries |
| Athena | PROD |  |  | prod\_countries |
| Athena | QA |  |  | qa\_canibalizacion |
| Athena | PROD |  |  | prod\_ canibalizacion |

## Tablas postgres

|  |  |
| --- | --- |
| **Schema** | **Tabla** |
| country\_<pais> | blocks |
| country\_<pais> | pois\_comercios\_servicios |
| country\_<pais> | view\_blocks |
| country\_<pais> | pois |
| country\_<pais> | view\_blocks |
| country\_<pais> | view\_gastos |
| country\_<pais> | view\_blocks |
| country\_<pais> | categories |
| country\_<pais> | income\_levels |

# Dns configurados [ ]

* <https://canibalizacion.geo-research.com/>
* https://canibalizacion -dev.geo-research.com/
* https://canibalizacion -qa.geo-research.com/

# Endpoints

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| utilidad | api | name | order | endpoint | verb |
| Status stepfunctions |  | Creada por Francisco Mendoza para consultar genéricamente estos estados |  | https://5xvveqna8d.execute-api.us-east-1.amazonaws.com/dev/state | POST |

# Api keys [ ]

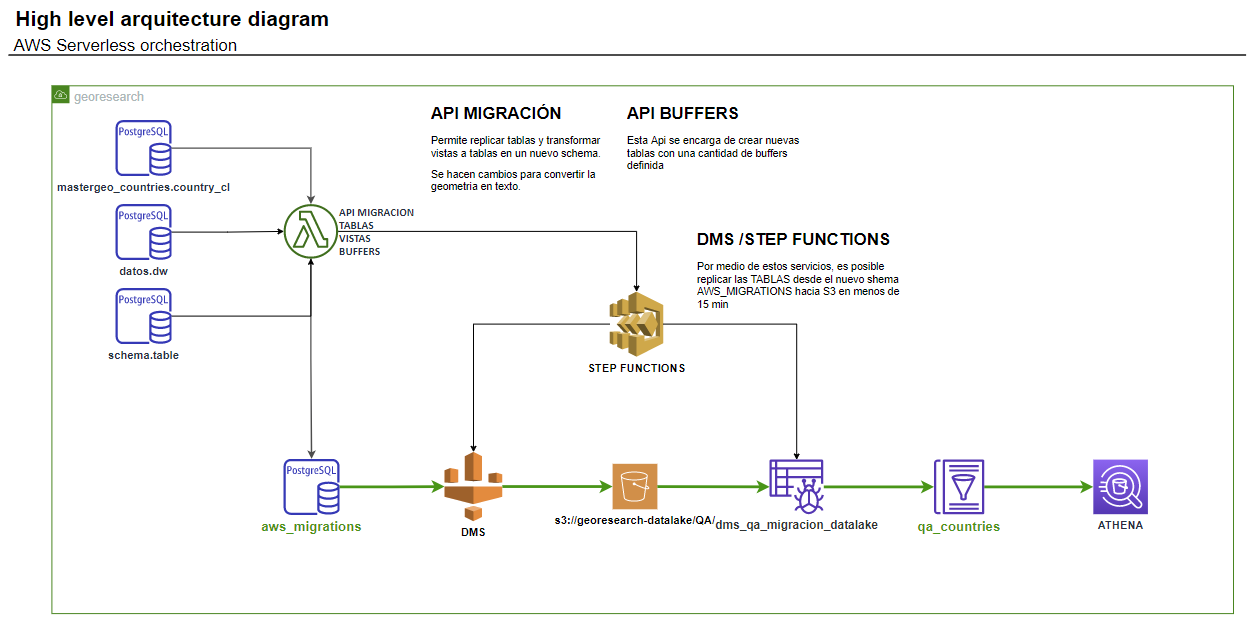
|  |  |
| --- | --- |
| **api key** | **clave** |
| canibalizacion-dev | HmtXfTvefC7QDpsSHQvvn2LFk4Yd19456mJc9Lei |
| canibalizacion -qa | bEfporJGeq88yRJy4vGcd3WdQzjwWzRaBEoYzzR1 |
| canibalizacion -prod | Fw60Wah3W15GtaF6yf7ok9XfoHWJT4642cDbbEUp |

# Repositorio [ ]

1. https://github.com/cvargas-geo/canibalizacin

# 

# Diagrama de arquitectura API DMS



## ~~Migración con step Functions~~ Se incluye en el servicio replicate-tables

Una vez que la creación de objetos el schema aws\_migrations esta validado, es momento de migrar las tablas a Amazon, para ello debe crear la solicitud input para el servicio creado con step Functions.

Esta solicitud deberá indicar el origen y destino de los datos a migrar, en el caso de origen deberá especificar el schema y tablas a migrar para esto el servicio permite agregar caracteres wilcard para buscar las tablas, por ejemplo, si se requiere migrar todas las tablas que comiencen con “country\_uy\_” se deberá especificar de la siguiente forma: “country\_uy\_%”

Dentro de la clave “TableMappings” se deberán especificar las reglas de mapeo, pero hay que tener en cuenta los siguientes puntos:

* Cada regla debe poseer un rule-id y rule-name único dentro de la solicitud
* Se debe indicar el schema de origen
* Indicar el carácter que coincida con las tablas a migrar, usar “**country\_%\_%**” para migrar todo.
* Se permiten agregar reglas de inclusión o exclusión mediante “rule-action”

"TableMappings": {

"rules": [

{

"rule-type": "selection",

"rule-id": "995195053",

"rule-name": "995195053",

"object-locator": {

"schema-name": "aws\_migrations",

"table-name": "country\_uy\_%"

},

"rule-action": "include",

"filters": []

}

]

}

### Detalle de la solicitud:

* SKIP: true/false Creado para etapa de desarrollo permite saltar algunos pasos del proceso como la creación de la instancia de replicación con el fin de hacer pruebas.
* SKIP\_DELETE : enable/vacio, permite saltar la etapa de eliminar la instancia de replicación para reutilizarla en pruebas.
* DEFAULT\_WAIT\_SECONDS :60 Indica cuando esperar entre cada etapa del proceso
* SOURCE\_ENDPOINT\_ARN : Indica arn del el endpoint origen creado para obtener datos
* TARGET\_ENDPOINT\_ARN: Indica arn del el endpoint destino creado para almacenar datos
* ReplicationInstanceIdentifier : Nombre asignado a la instancia de replicación, este debe se único.
* ReplicationTaskIdentifier: Nombre asignado a la tarea de replicación, este debe se único.
* FILTER\_INSTANCE\_REPLICATION: Lista que contiene el nombre de la instancia de replicación para obtener el detalle de la ejecución posteriormente que se ejecute.
* FILTER\_TASK\_REPLICATION: Lista que contiene el nombre de la tarea de replicación para obtener el detalle de la ejecución posteriormente que se ejecute.
* REPLICATION\_INSTANCE\_PARAMETERS : Parámetros para crear la instancia de replicación

{

"SKIP": "enables",

"SKIP\_DELETE": "enables",

"DEFAULT\_WAIT\_SECONDS": 60,

"SOURCE\_ENDPOINT\_ARN": "arn:aws:dms:us-east-1:958531303673:endpoint:452KHXMN3SCETMQC75H3GZMR4TOJE4FJ35U5CFA",

"TARGET\_ENDPOINT\_ARN": "arn:aws:dms:us-east-1:958531303673:endpoint:LXF5MWFWSY7M7US6US4KPDME242KSJSLFBGNOBQ",

"ReplicationInstanceIdentifier": "dms-replication-instance-for-datalake",

"ReplicationTaskIdentifier": "dms-replication-task-for-datalake",

"FILTER\_INSTANCE\_REPLICATION": [

"dms-replication-instance-for-datalake"

],

"FILTER\_TASK\_REPLICATION": [

"dms-replication-task-for-datalake"

],

"REPLICATION\_INSTANCE\_PARAMETERS": {

"ReplicationInstanceIdentifier.$": "$.ReplicationInstanceIdentifier",

"ReplicationInstanceClass": "dms.c5.4xlarge",

"ReplicationInstanceClassMin": "dms.t2.micro",

"AllocatedStorage": 50,

"EngineVersion": "3.4.6",

"MultiAZ": false,

"ReplicationSubnetGroupIdentifier": "default-vpc-f944259d",

"VpcSecurityGroupIds": [

"sg-bd8547c4"

]

}

"MIGRATION\_TASK\_PARAMETERS": {

"ReplicationTaskIdentifier.$": "$.ReplicationTaskIdentifier",

"ReplicationInstanceArn.$": "$.DescribeResult.ReplicationInstances[0].ReplicationInstanceArn",

"SourceEndpointArn.$": "$.SOURCE\_ENDPOINT\_ARN",

"TargetEndpointArn.$": "$.TARGET\_ENDPOINT\_ARN",

"MigrationType": "full-load",

"TableMappings": {

"rules": [

{

"rule-type": "selection",

"rule-id": "995195053",

"rule-name": "995195053",

"object-locator": {

"schema-name": "aws\_migrations",

"table-name": "country\_uy\_%"

},

"rule-action": "include",

"filters": []

}

]

}

}

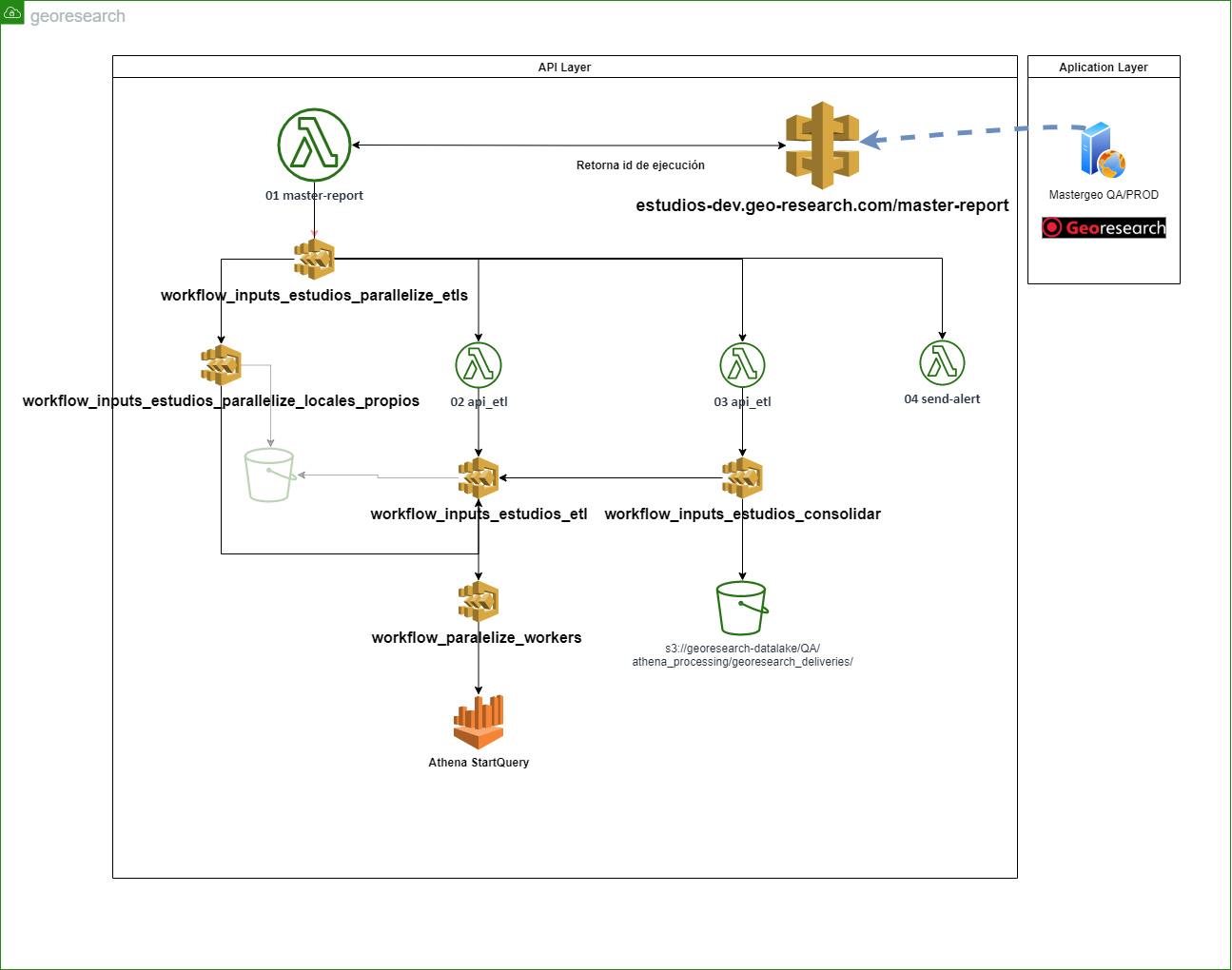
Nota: EngineVersion se actualiza a 3.4.6

Problemas detectados

* 3.4.4: Deprecado por el servicio DMS
* 3.4.5: Las tablas quedan sin headers en Athena.

# Diagrama de la arquitectura api estudios [ ]

Toda la infraestructura planteada fue migrada desde un proceso en Airflow – gcp hacia AWS utilizando Lambda, Athena, S3, Step Functions y ApiGateway.



## Lambda

Actualmente las siguientes Lambdas están desplegadas en la cuenta productiva AWS, su acceso se debe solicitar al JP respectivo.

1. etl-estudios-2022-dev-master-report

## Endpoint

Para hacer consultas se habilito la siguiente url que puede ser consultada con Postman (www.postman.com) para hacer pruebas.

* POST - https://estudios-dev.geo-research.com/master-report

## Salida de resultados

Una vez terminado el proceso iniciado por el servicio master-report este creara los archivos final en la siguiente ubicación de S3 , en este caso **new\_customer** corresponde al nombre asignado para la solicitud de ejemplo.

Ejemplo para ambiente de QA

s3://georesearch-datalake/QA/athena\_processing/georesearch\_deliveries/**new\_customer\_inputs**

Ejemplo para ambiente de PROD

s3://georesearch-datalake/PROD/athena\_processing/georesearch\_deliveries/**new\_customer\_inputs**

## Solicitud

Para realizar pruebas utilizar el siguiente input, en rojo parámetros obligatorios y en azul opcionales.

### ~~Versión anterior~~

**{**

**"customer\_list": [**

**{**

**"drop\_workflow": false,**

**"report\_to": [**

**"cvargas@georesearch.cl"**

**],**

**"customer\_name": "cvargas\_inputs",**

**"country\_list": [**

**"mx"**

**],**

**"****buffer\_list": [**

**1000 , 2000 , 5000**

**],**

**"id\_gastos": [**

**1,**

**2,**

**3**

**],**

**"etl\_list": [**

**"demografico",**

**"gastos",**

**"competencias"**

**],**

**"parametros": {**

**"competencias": {**

**"pois\_state\_id": 1,**

**"substring\_id": [**

**279,**

**1096,**

**1445**

**]**

**},**

**"locales\_propios": {**

**"pois\_state\_id": 1,**

**"substring\_id": [**

**279**

**],**

**"comunas\_mx":["CIUDAD DE MEXICO"],**

**"comunas\_cl":["SANTIAGO"]**

**}**

**}**

**}**

**]**

**}**

### Ultima versión

Lo destacado corresponde al nuevo formato de parámetros para el servicio, los colores de las claves se siguen respetando.

Nota: se incluye la subclave environment

{

"environment": "PROD",

"customer\_list": [

{

"drop\_workflow": false,

"customer\_name": "new\_customer",

**"report\_to": [**

**"new\_customer@georesearch.cl"**

**],**

"country\_list": [

"ar",

"uy"

],

"buffer\_list": [

1000

],

"etl\_list": [

"demografico",

"gastos",

"competencias"

],

"parametros": {

"competencias": {

"uy": {

"pois\_state\_id": 1,

"substring\_id": [

279,

1096,

1445

],

"category\_id": [

1,

2

]

},

"ar": {

"pois\_state\_id": 1,

"substring\_id": [

279,

1096,

1445

],

"category\_id": [

1,

2

]

}

},

"locales\_propios": {

"uy": {

"pois\_state\_id": 1,

"substring\_id": [

279,

1096,

1445

]

},

"ar": {

"pois\_state\_id": 1,

"substring\_id": [

279,

1096,

1445

]

}

},

"gastos": {

"uy": {

"id\_gastos": [

1,

2

]

},

"ar": {

"id\_gastos": [

1,

2

]

}

},

"demografico": {

"uy": {

"blocks": [

1,

2

],

"administrative\_area\_level\_1": ['METROPOLOTANA'],

"administrative\_area\_level\_2": ['SANTIAGO'],

"administrative\_area\_level\_3": ['MAIPU'],

},

"ar": {

"blocks": [

1,

2

]

}

}

}

}

]

}

### Ejemplo 2

{

"environment": "PROD",

"customer\_list": [{

"drop\_workflow": false,

"customer\_name": "example\_customer",

"report\_to": [

"example\_customer@georesearch.cl"

],

"country\_list": [

"cl"

],

"buffer\_list": [

100

],

"etl\_list": [

"demografico",

"gastos",

"competencias"

],

"parametros": {

"competencias": {

"cl": {

"pois\_state\_id": 1,

"substring\_id": [

279,

1096,

1445,

1552

],

"category\_id": [

279,

1096,

1445,

1552

]

}

},

"locales\_propios": {

"cl": {

"pois\_state\_id": 1,

"substring\_id": [

279,

1096,

1445,

1552

]

}

},

"gastos": {

"cl": {

"id\_gastos": [

1,

2,

3

]

}

},

"demografico": {

"cl": {

"administrative\_area\_level\_1": ["METROPOLITANA"],

"administrative\_area\_level\_2": ["SANTIAGO"],

"administrative\_area\_level\_3": ["MAIPU"],

"block\_id": [

1104010104,

1104010105,

1104010106,

1104010110,

1104010111,

1104010122

]

}

}

}

}]

}

# Lista de etls disponibles

La salida de resultados contempla la siguiente información del negocio:

1. Datos demográficos (base)
2. Datos de competencias (base)
3. Datos de gatos (base)
4. Datos de atractores comerciales(base)
5. Locales propios (opcional)

# Diccionario de claves:

Definiciones previas

Objeto: {}

Lista: []

## customer\_list (requerido) [{},{}…{}]

Lista de solicitudes que se compone de una o más solicitudes.

Ejemplo:

{“customer\_list”:[{solicitud\_1}…{ solicitud\_2}]}

## solicitud

Objeto que contiene los destalles enviados en el Excel, va incorporado dentro de su clave padre:“customer\_list”

Parámetros de la solicitud, en rojo campos requeridos celeste opcionales:

* **customer\_name**: requerido, string, nombre de la solicitud que se asocia al cliente.
* **enviroment**:requerido,string indica el entorno de trabajo de los datos.
* **country\_list**: requerido, lista de países en formato ISO, contiene 1 o más. Ejemplo: [“cl”…”mx”] , actualmente solo permite los siguientes códigos isos: cl,mx,co,pe.
* **buffer\_list**: requerido, lista de números que contiene los buffers requeridos en metros, contiene 1 o más, ejemplo [1000,2000,3500], actualmente solo permite los siguientes: [500,600,1000,1500,2000,2500,3000,3500,4000,4500,5000]
* **id\_gastos**: requerido, lista de números que contiene los ids de los substring requeridos para los gastos, puede contener 1 o más, ejemplo [2,26].
* **etl\_list**: requerido, lista de strings que contiene los nombres de los etl a solicitar, puede contener 3 o más ejemplo [“demografico”,”gastos”,”competencias”].
* **parametros**: requerido, objeto que contiene los requerimientos de cada etl, se detalla más abajo. Ejemplo

{

        "competencias": {},

"locales\_propios": {}

}

* **drop\_workflow**: opcional, boleano, indicador para borrar los objetos internamente creados de una solicitud. Ejemplo true o false, (no acepta mayúsculas).
* **report\_to**: opcional, lista de correos a los que serán enviada una notificación cuando el proceso finalice ejemplo: [“customer@georesearch.com”]

### Filtro de regiones provincias y comunas

Este filtro por ahora contempla una sola regla la cual debe ser especificada en los 3 niveles para evitar colisiones. Ejemplo: para filtrar solo datos demográficos de la comuna de MAIPU de deben especificar además la provincia de SANTIAGO y la región Metropolitana. Esto solo aplica para la subclave demográfico.

Captura de pantalla de computadora

Descripción generada automáticamente

### Subclave parámetros

Esta clave está destinada a especificar al detalle cada parámetro necesario del etl.

Para identificarlas se agregan según el nombre etl a que hacen referencia.

Por ejemplo, en el caso de las competencias y locales propios se debe especificar la lista de sub strings y su estado.

Como locales propios es un etl opcional, la subclave dentro de parámetros también lo es, pero en el caso de competencias no, pues es un etl obligatorio.

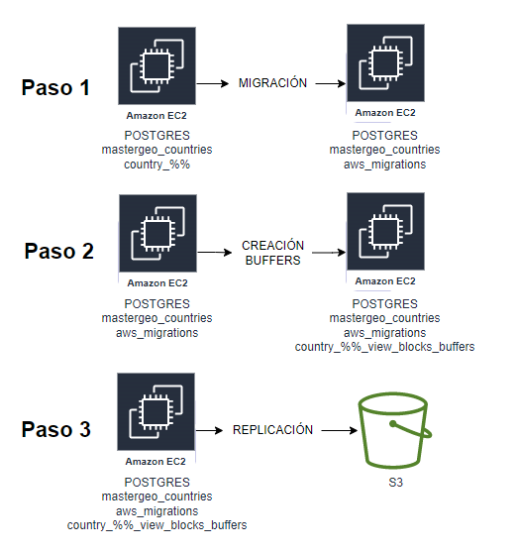
## ~~Futuros cambios~~ ya incluidos

Estos serán implementados para mejorar la actual solicitud de inputs.

1. Se incluirá la clave “country” dentro de “parametros” para especificar parámetros por países, esto también aplicará para id\_gastos que será movido dentro de parámetros, Esto se explica pues lo id gastos de un país pueden ser diferentes en otro.

# Flujo de actualización de datos

A continuación, se mencionan los pasos principales para actualizar las tablas en Athena para la generación de los inputs estudios.



1. Migración: Consiste en generar la solicitud para un conjunto de tablas/vistas y migrarlas al schema aws\_migrations, en este proceso asíncrono se convierten las columnas geometry a texto y las vistas a tablas por lo que se debe verificar en la bbdd los objetos creados. (postgres-postgres)
2. Creación de buffers: Generar la solicitud síncrona para la creación de los buffers para un conjunto de tablas, tiende a tardar menos de 15 min. (postgres-postgres)
3. Replicación: Consiste en tomar los objetos de aws\_migrations y llevarlos a S3 mediante una solicitud asíncrona, esta solicitud posee un conjunto de reglas de replicación que permitirán al servicio DMS migrar las tablas. ( postgres - aws ) sf : workflow\_dev\_dms\_replicacion

## Paso 1 Migración

Si por ejemplo tenemos una actualización para chile o cualquier otro país(mx,co,pe,ar,uy), las siguientes tablas serán necesarias para realizar el proceso:

1. country\_cl.pois\_comercios\_servicios\_view
2. country\_cl.view\_blocks
3. country\_cl.empresas
4. country\_cl.view\_gastos
5. country\_cl.categories
6. country\_cl.canasta\_categoria
7. country\_cl.canastas\_total
8. country\_cl.income\_levels

Con esto en mente se deberá preparar el request para la api, con el objetivo de mover las tablas al esquema de migración “**aws\_migrations**“, luego desde aquí otro servicio se encargara de copiar los datos a S3.

**Endpoint**: https://estudios-qa.geo-research.com/migrate-tables

A continuación, el siguiente request json será utilizado para completar el primer paso. La clave environment podrá ser PROD o QA.

{

"body": {

"environment": "PROD",

"tables": [

{

"schema": "country\_cl",

"table": "pois\_comercios\_servicios\_view"

},

{

"schema": "country\_cl",

"table": "view\_blocks"

},

{

"schema": "country\_cl",

"table": "empresas"

},

{

"schema": "country\_cl",

"table": "view\_gastos"

},

{

"schema": "country\_cl",

"table": "categories"

},

{

"schema": "country\_cl",

"table": "canasta\_categoria"

},

{

"schema": "country\_cl",

"table": "canastas\_total"

},

{

"schema": "country\_cl",

"table": "income\_levels"

}

]

}

}

Nota: Para abarcar más de un país, el siguiente script en Python ayudara a generar el json anterior, solo es necesario incluir los schemas necesarios mientras estén homologados. (MX,CL,PE,CO)(UR y AR en proceso)

Consulta para verificar el estado de las tablas

/\*PARA USAR EN DBEAVER: consulta por los registros que faltan por copiar \*/

WITH

SOURCE\_TABLES AS (

SELECT COUNT(\*) AS RECUENTO , 'country\_cl.pois\_comercios\_servicios\_view' AS TABLA FROM country\_cl.pois\_comercios\_servicios\_view UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.view\_blocks' AS TABLA FROM country\_cl.view\_blocks UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.empresas' AS TABLA FROM country\_cl.empresas UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.view\_gastos' AS TABLA FROM country\_cl.view\_gastos UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.categories' AS TABLA FROM country\_cl.categories UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.canasta\_categoria' AS TABLA FROM country\_cl.canasta\_categoria UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.canastas\_total' AS TABLA FROM country\_cl.canastas\_total UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.income\_levels' AS TABLA FROM country\_cl.income\_levels

)

,TARGET\_TABLES AS (

SELECT COUNT(\*) AS RECUENTO , 'country\_cl.pois\_comercios\_servicios\_view' AS TABLA FROM aws\_migrations.country\_cl\_pois\_comercios\_servicios\_view UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.view\_blocks' AS TABLA FROM aws\_migrations.country\_cl\_view\_blocks UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.empresas' AS TABLA FROM aws\_migrations.country\_cl\_empresas UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.view\_gastos' AS TABLA FROM aws\_migrations.country\_cl\_view\_gastos UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.categories' AS TABLA FROM aws\_migrations.country\_cl\_categories UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.canasta\_categoria' AS TABLA FROM aws\_migrations.country\_cl\_canasta\_categoria UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.canastas\_total' AS TABLA FROM aws\_migrations.country\_cl\_canastas\_total UNION ALL SELECT COUNT(\*) AS RECUENTO , 'country\_cl.income\_levels' AS TABLA FROM aws\_migrations.country\_cl\_income\_levels

)

SELECT

A.TABLA ,

CASE WHEN A.RECUENTO - B.RECUENTO >= 0 THEN A.RECUENTO - B.RECUENTO

ELSE -1 END AS POR\_COPIAR

FROM SOURCE\_TABLES A

LEFT JOIN TARGET\_TABLES B ON A.TABLA = B.TABLA

ORDER BY 2 DESC

Script de generación

import json

environment = 'PROD'

selected\_countries = [

'country\_cl' ,

# 'country\_mx' ,

# 'country\_co' ,

# 'country\_pe' ,

# 'country\_ar',

# 'country\_uy'

]

tables = [

"pois\_comercios\_servicios\_view",

"view\_blocks",

"empresas",

"view\_gastos",

"categories",

"canasta\_categoria",

"canastas\_total",

"income\_levels"

]

limit =''

migracion\_tablas = {

"environment": "PROD",

"tables": [

{

"schema": country,

"table": table,

} for table in tables for country in selected\_countries

]

}

request = {"body":migracion\_tablas}

print(json.dumps( request , indent=4))

Para ejecutar el código se puede ejecutar en los siguientes enlaces:

* <https://jupyter.org/try-jupyter/lab/>
* <https://replit.com/languages/python3>

## Paso 2 Creación de buffers

Dado que es los buffers se cran a partir de las tablas view\_blocks o pois , también será necesario actualizarlas por lo que se debe preparar la siguiente solicitud .

Endpoint: https://estudios-qa.geo-research.com/buffer

{

"body": {

"environment": "PROD",

"schemas": [

"country\_cl"

],

"tables": [

"view\_blocks"

],

"buffers": [

100,

500,

600,

800,

1000,

1500,

2000,

2500,

3000,

3500,

4000,

4500,

5000

]

}

}

Consulta de validación de datos:

/\*PARA USAR EN DBEAVER : consulta por los registros que faltan por copiar \*/

WITH

SOURCE\_TABLES AS (

SELECT COUNT(\*) AS RECUENTO , 'country\_cl.view\_blocks' AS TABLA FROM country\_cl.view\_blocks

)

,TARGET\_TABLES AS (

SELECT COUNT(\*) AS RECUENTO , 'country\_cl.view\_blocks' AS TABLA FROM aws\_migrations.country\_cl\_view\_blocks\_buffers

)

SELECT

A.TABLA ,

CASE WHEN A.RECUENTO - B.RECUENTO >= 0 THEN A.RECUENTO - B.RECUENTO

ELSE -1 END AS POR\_COPIAR

FROM SOURCE\_TABLES A

LEFT JOIN TARGET\_TABLES B ON A.TABLA = B.TABLA

ORDER BY 2 DESC

Consulta para agregar una columna buffer de ayuda

ALTER TABLE aws\_migrations.country\_mx\_view\_blocks\_buffers

ADD buffer\_5000 text NULL ;

UPDATE aws\_migrations.country\_mx\_view\_blocks\_buffers A

SET

buffer\_5000 = st\_astext(st\_buffer(st\_setsrid(st\_point(B.longitud, B.latitud), 4326)::geography, 5000)::geometry)

FROM country\_mx.view\_blocks B

WHERE A.id = B.id;

Consulta para crear la tabla de buffers

DROP TABLE IF EXISTS aws\_migrations.country\_mx\_view\_blocks\_buffers ;

CREATE TABLE aws\_migrations.country\_mx\_view\_blocks\_buffers AS

SELECT

id ,

block\_id,

latitud,

longitud,

administrative\_area\_level\_1,

administrative\_area\_level\_2

FROM country\_mx.view\_blocks ;

## Paso 3 Replicación

Los objetos de aws\_migrations son copiados a S3 y Athena, mientras se agreguen las reglas de mapeo necesarias en la solicitud.

**Endpoint**: https://estudios-qa.geo-research.com/replicate-tables

Nota: El servicio DMS permite el uso del wildcard “%” para realizar la búsqueda de los objetos.

{

"body": {

"environment": "PROD",

"rules": [

{

"schema-name": "aws\_migrations",

"table-name": "country\_cl\_% "

}

]

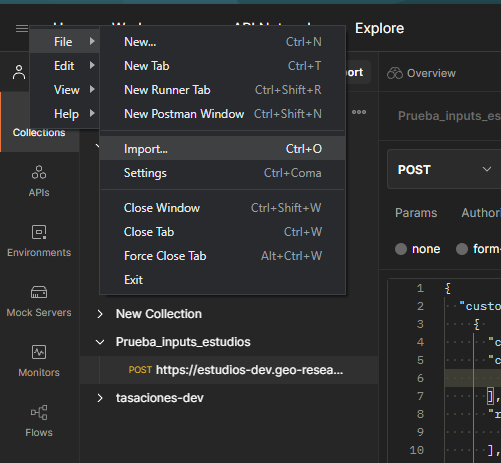
}

}

Para verificar el estado del proceso se deberán ver los logs de la stepfunction workflow\_dev\_dms\_replicacion

# Postman

Utilidades: https://jsoneditoronline.org/#/

1. Importar el json al programa File/Import
2. 
3. Interfaz de usuario gráfica, Aplicación

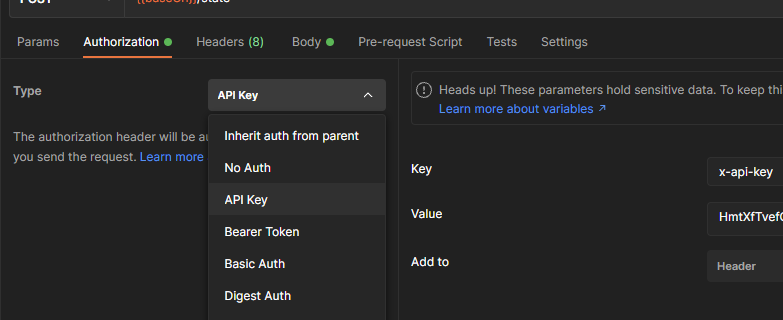
   Descripción generada automáticamente
4. Hay que asegurar que los headers estén activados.Interfaz de usuario gráfica, Aplicación

   Descripción generada automáticamente
5. Finalmente hacer una prueba, en la pestaña body + botón SEND
6. Interfaz de usuario gráfica, Texto

   Descripción generada automáticamente
7. Esto no indicara nada más, pero habilitare una opción para indicar fallos si la solicitud quedo malformada.

### Agregar API KEY

En este caso para incluirla en la request de postman agregarla en authorization / type API Key / y setar el campo value.



# Actualizaciones de de la api para inputs a medida

Dentro de los 2 casos de uso tenemos que cuando se requiere input con una modificación muy particular será necesario un desarrollo del flujo del etl , para ello solo será necesario crear una nueva consulta para el etl /stage particular que se necesite , pero deberá agregarse la solicitud dentro de siguiente archivo para que esta pueda ser reflejada en los resultados finales.

En este ejemplo se simula que la nueva solicitud cambia en todas sus consultas, lo que se va indicando dentro de la clave stages.

Texto

Descripción generada automáticamente