Tarea 2 de Algoritmo de ruteo y redes resilientes

|  |  |  |
| --- | --- | --- |
| *Integrantes* | *:* | *David Salas* |
| *Sección* | *:* | *1* |
| *Semestre* | *:* | *2023-2* |
| *Profesor* | *:* | *Nicolas Boettcher* |

|  |  |  |
| --- | --- | --- |
| *Fecha de A* | *:* | *04/12/2023* |

Todos los archivos utilizados se pueden ver en este github:

[Tarea2\_Github](https://github.com/DavidSalas97/Tarea2ARYRR)

**Descargar infraestructura**

Para esta parte se descarga la infraestructura a utilizar, que en este caso es la de fibra óptica, en donde se descarga directamente del canvas en modulos, el archivo “PortalTransparencia\_Yafun.zip”, luego descomprimimos lo que tiene al interior y obtenemos cinco archivos que son:

* Fibra\_optica\_detectada.dbf
* Fibra\_optica\_detectada.prj
* Fibra\_optica\_detectada.qpj
* Fibra\_optica\_detectada.shp
* Fibra\_optica\_detectada.shx

**Creación de la tabla con la información**

Luego, creamos un archivo **Dockerfile** que contiene Postgres:16 como imagen y las instalaciones de todo lo necesario, tanto de los documentos mencionado anteriormente, como la instalación de postgres16, postgist16 y pgrouting.

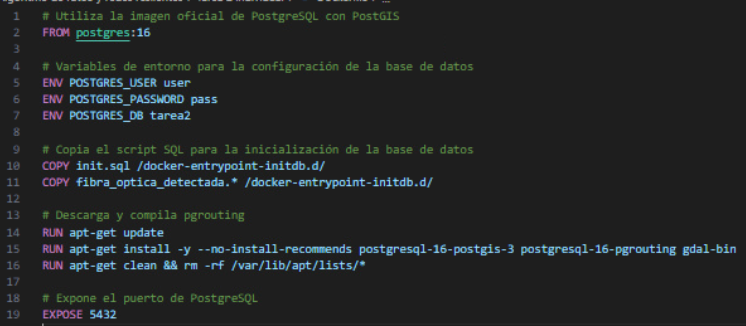


Figura 1: Archivo “Dockerfile”

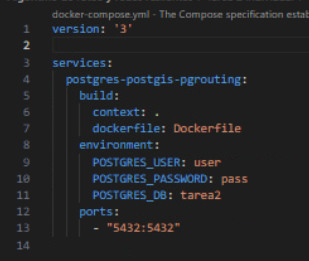


Figura 2: Archivo “docker-compose.yml”

Información del archivo init.sql, para que cree la tabla requerida para este trabajo

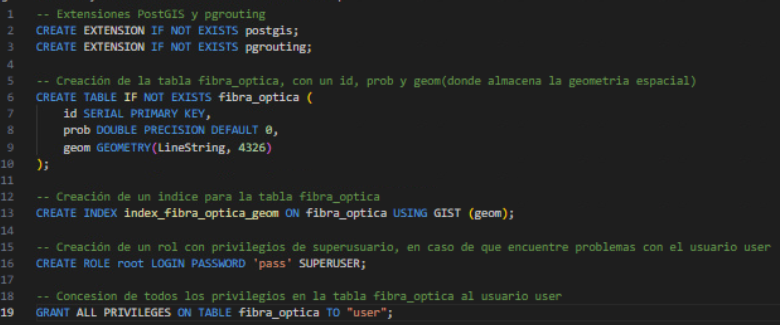


Figura 3: Archivo “init.sql”

Donde tenemos las extensiones de postgis y pgrouting, la creación de la tabla a utilizar, un index para el id de la tabla “fibra\_optica”, y por ultimo un rol de super usuario y conceder todos los privilegios al usuario de “user”.

Luego de haber ejecutado el dockerfile, con el siguiente comando:

**Docker compose build**

**Docker compose up**

Tenemos lo siguiente:

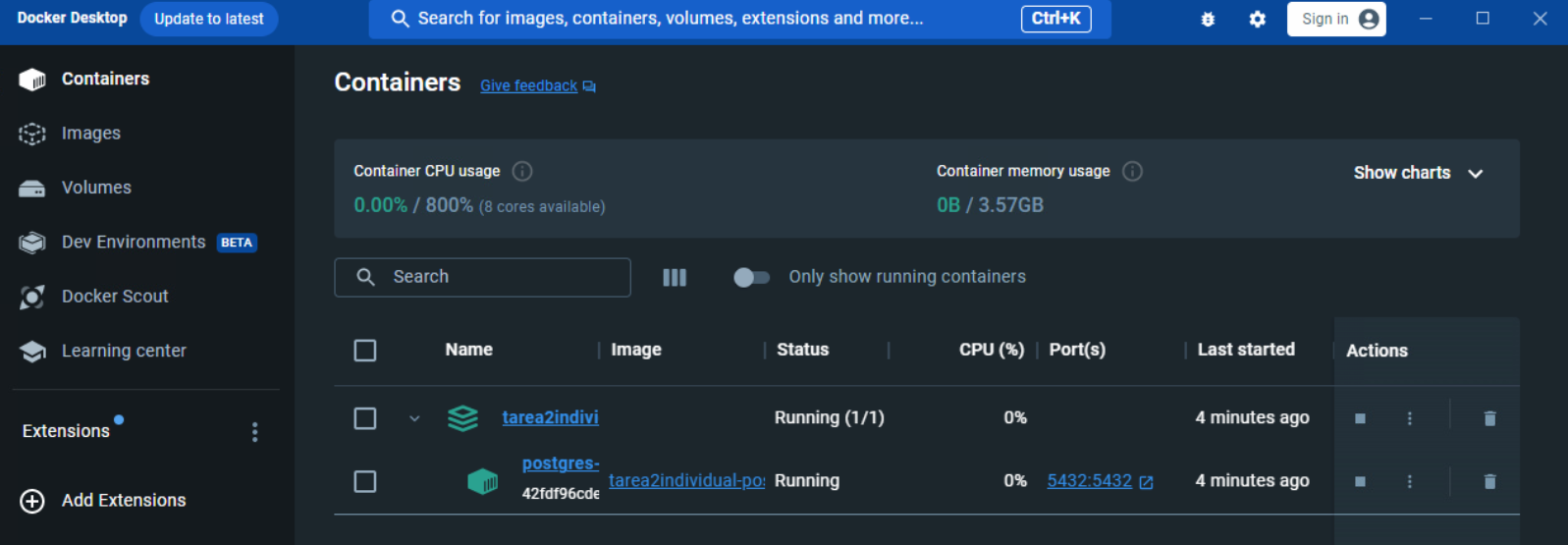


Figura 4: El contenedor levantado

En donde podemos visualizar que los contenedores creados por el dockerfile están levantados, pero aún no tenemos la información de la fibra\_optica\_detectada.

Para esto utilizaremos una herramienta llamada **“ogr2ogr”**, que nos sirva para transformar archivo de tipo SHP, en archivo que nosotros queramos, en este caso, sería de transformarlo a un formato de la base de datos SQL para adquirir sus datos a la tabla creada “fibra\_optica”. Todo esto dentro de la bash del contenedor, con el siguiente comando podemos entrar a la bash de este:

**docker exec -it 42fdf96cdec3 bash**

En donde, el identificador “**42fdf96cdec3**”, sería el id del contenedor usado.

Con la siguiente línea de comando podemos hacer lo mencionado anteriormente:

**ogr2ogr -f "PostgreSQL" PG:'dbname=tarea2 user=user password=pass' -nln fibra\_optica -lco GEOMETRY\_NAME=geom -append fibra\_optica\_detectada.shp**

Donde el resultado entregado es el siguiente:

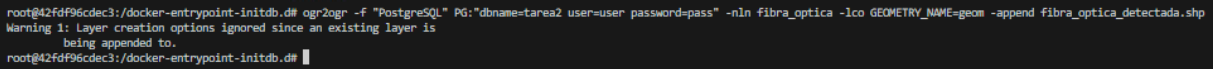


Figura 5: Comprobación comando ogr2ogr

Cabe destacar que, para que ejecute como debe el comando, este debe encontrarse con el archivo “fibra\_optica\_detectada.shp” en el mismo destino, por se ingresa al archivo “docker-entrypoint-initdb.d”.

Una vez realizado esto, procedemos a verificar si existe la tabla y sus datos correspondiente de lo realizado anteriormente, con el siguiente comando en la bash del contenedor, entramos a PostgreSQL:

**psql -U user -d tarea2**

Luego, con el comando dentro de psql, “***\dt”,*** podemos visualizar las tablas que hay dentro.

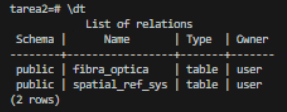


Figura 6: Verificación de tablas

Y con este comando sabemos si posee los datos que les añadimos con el comando ogr2ogr:

**SELECT COUNT(\*) FROM fibra\_optica;**

Con el siguiente resultado:

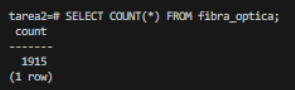


Figura 7: Comprobación de datos

**Conversión a red**

Una vez, ya realizado todo lo mencionado anteriormente, se procede a la conversión a red, para hacer uso de pgrouting.

En donde, se hace unos cambios a la tabla “fibra\_optica”, para poder convertirla en una topología red, que ente caso creamos dos columnas, que serian el inicio y final, que se representaran como nodos de inicio y final, valga la redundancia, para cada segmento de la red

**ALTER TABLE fibra\_optica ADD COLUMN inicio integer;**

**ALTER TABLE fibra\_optica ADD COLUMN final integer;**

Una vez agregadas las tablas, se realiza el siguiente comando de SQL, para agregarle valores correctos a los nodos mencionado anteriormente.

**UPDATE fibra\_optica SET**

**inicio = CAST(ST\_X(nodes.inicio) AS integer),**

**final = CAST(ST\_X(nodes.final) AS integer)**

**FROM (**

**SELECT**

**edge.id AS edge\_id,**

**edge.inicio,**

**edge.final**

**FROM (**

**SELECT**

**id,**

**ST\_StartPoint(geom)::geometry(Point, 4326) AS inicio,**

**ST\_EndPoint(geom)::geometry(Point, 4326) AS final**

**FROM fibra\_optica**

**) AS edge**

**JOIN (**

**SELECT**

**id,**

**ST\_StartPoint(geom)::geometry(Point, 4326) AS inicio,**

**ST\_EndPoint(geom)::geometry(Point, 4326) AS final**

**FROM fibra\_optica**

**) AS nodes**

**ON edge.id = nodes.id**

**) AS nodes**

**WHERE fibra\_optica.id = nodes.edge\_id;**

Donde el resultado se puede ver de la siguiente Figura:

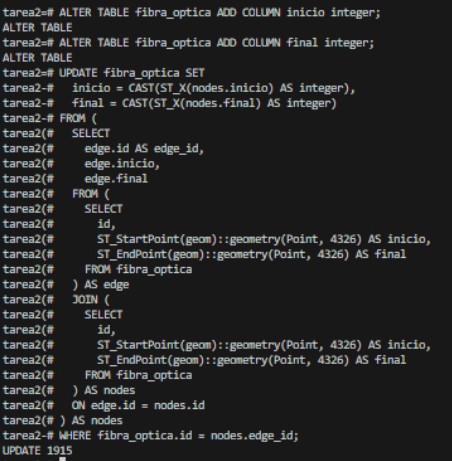


Figura 8: Comprobación de los datos agregados

Ya agregado estos nodos y configurados con los valores correspondiente, se procede a crear la topología con el siguiente comando:

**SELECT pgr\_createTopology('fibra\_optica', 0.0001, 'geom', 'id', 'inicio', 'final');**

En donde el resultado de crear esta topología se ve en la siguiente figura:

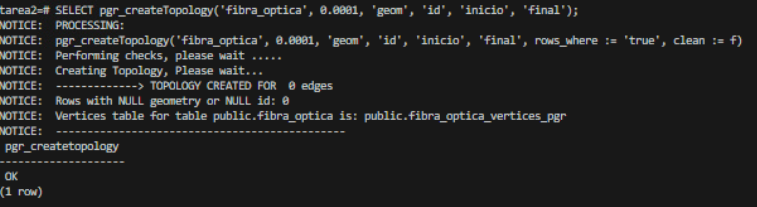


Figura 9: Creación de topología red con “pgr\_createTopology”

Debido a problemas de visualización para ver los datos, no puedo mostrar una figura con los datos ya realizado:

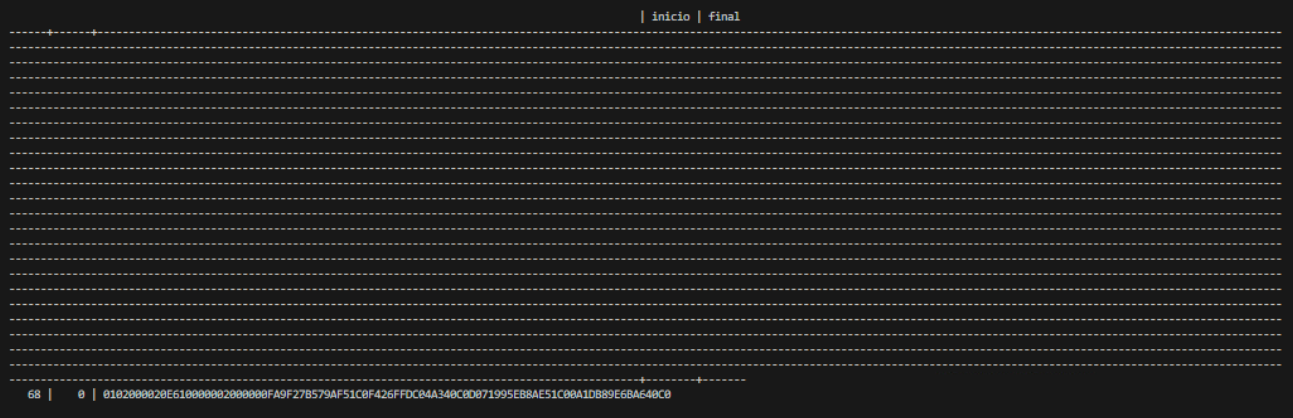


Figura 10: Mala visualización de los datos

**Script ATTR**

Para este caso el script de ATTR que utilizaremos es el siguiente:

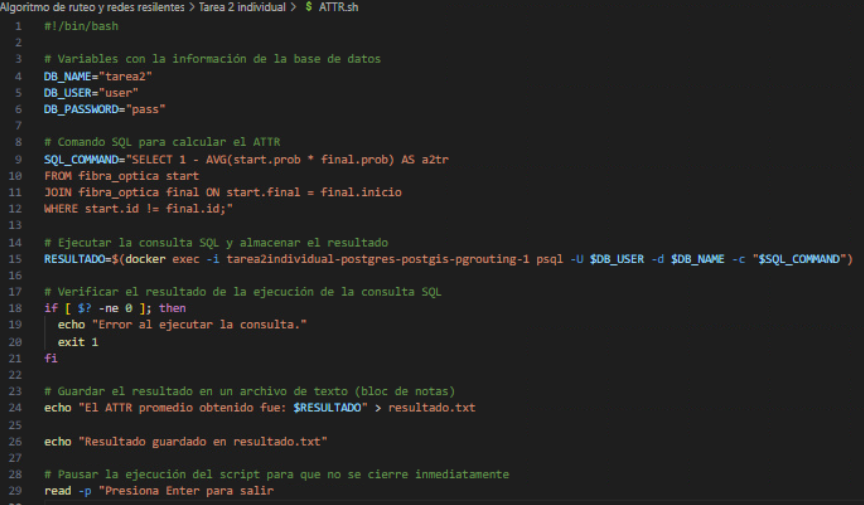


Figura 11: Script promedio ATTR

En donde, calculamos el ATTR promedio entre los nodos de inicio y final, luego el resultado obtenido de esto se guarda en un archivo llamado “resultado.txt”, donde nos dio el siguiente resultado:

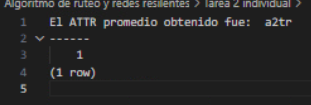


Figura 12: Resultado de aplicar ATTR promedio

Que, en este caso, podemos decir que lo obtenido fue un 100%, ya que este cálculo se realiza antes de la falla, donde tenemos que todas las probabilidades asignadas son de 0, dada la creación de la tabla sin valores.