



# ESPE

**UNIVERSIDAD DE LAS FUERZAS ARMADAS**  
**INNOVACIÓN PARA LA EXCELENCIA**

## **UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE**

**SISTEMA AVANZADO DE BASE DE  
DATOS**

**TEMA: EXAMEN UNIDAD UNO**

**ESTUDIANTE: ANDRANGO CARCHIPULLA CHRISTIAN JHONATAN**

**DOCENTE: ING. ALEXIS DARIO ESTEVEZ SALAZAR**

**NRC: 15031**

**SANGOLQUI**

***Noviembre 2023 - Marzo 2024***

## A. OBJETIVOS

- Implementar un diseño de base de datos en el que muestre el modelamiento y sus replicación en la base de datos de PostgreSQL.
- Detallar cada una de las implementaciones que se abordó en cada de los ítems planteados en la actividad.

## B. DESARROLLO

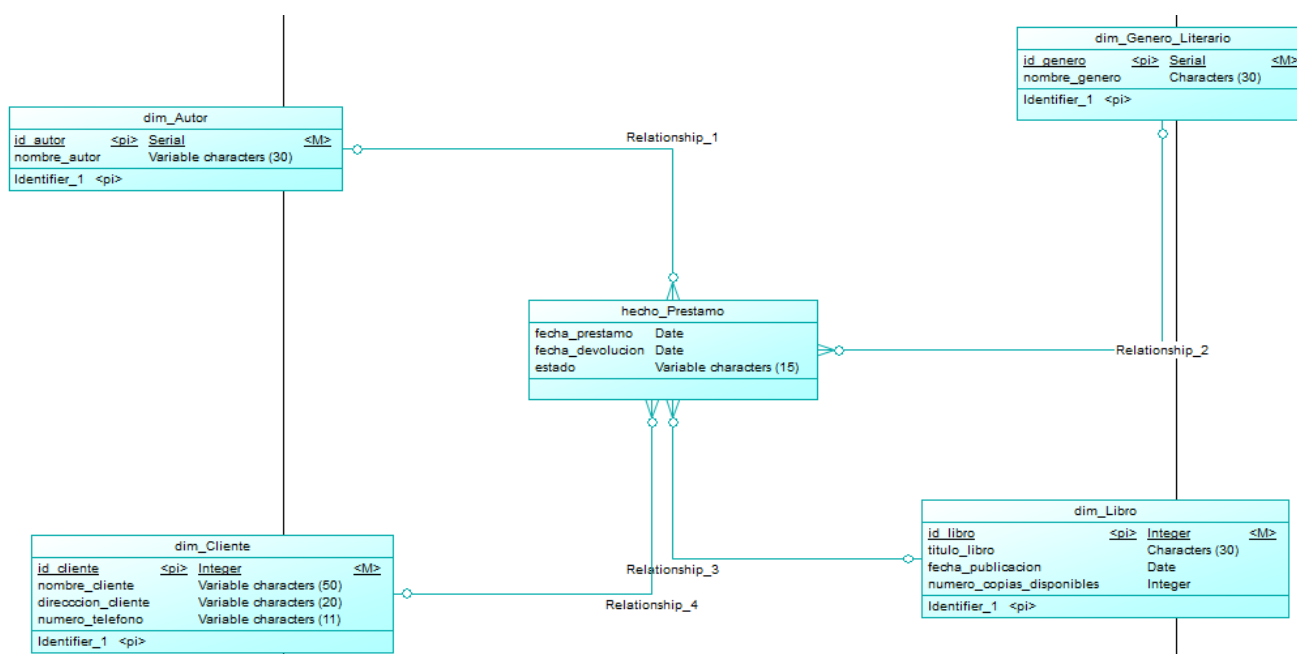
### Planteamiento

Una biblioteca desea implementar un sistema de gestión de información para organizar sus libros, autores, géneros y clientes. El sistema debe permitir a los bibliotecarios registrar nuevos libros, asignar autores y géneros a cada libro, realizar préstamos a los clientes y mantener un registro de los préstamos realizados. Además, se desea que la base de datos esté diseñada siguiendo el modelo de copo de nieve para garantizar la eficiencia y la integridad de los datos.

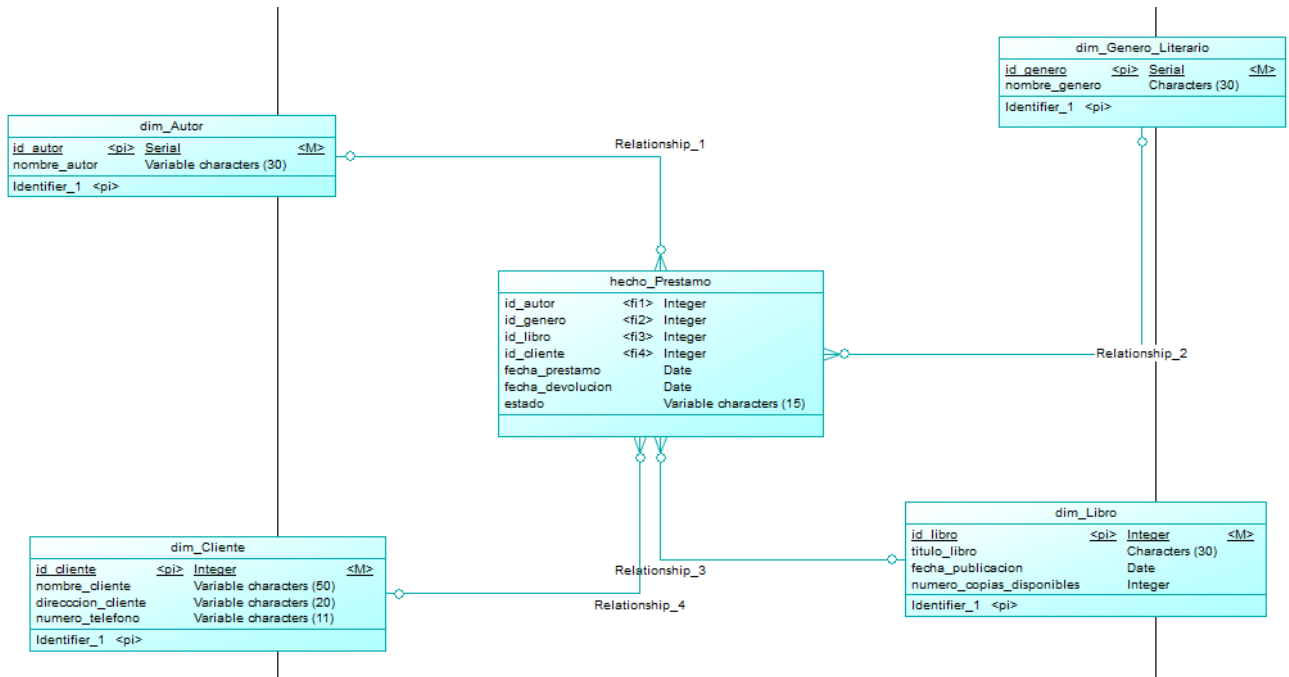
### Requisitos del Sistema:

- Los libros tienen un identificador único, un título, una fecha de publicación y un número de copias disponibles.
- Cada autor tiene un nombre único y puede escribir varios libros.
- Cada libro pertenece a uno o más géneros.
- Los clientes pueden realizar préstamos de libros.

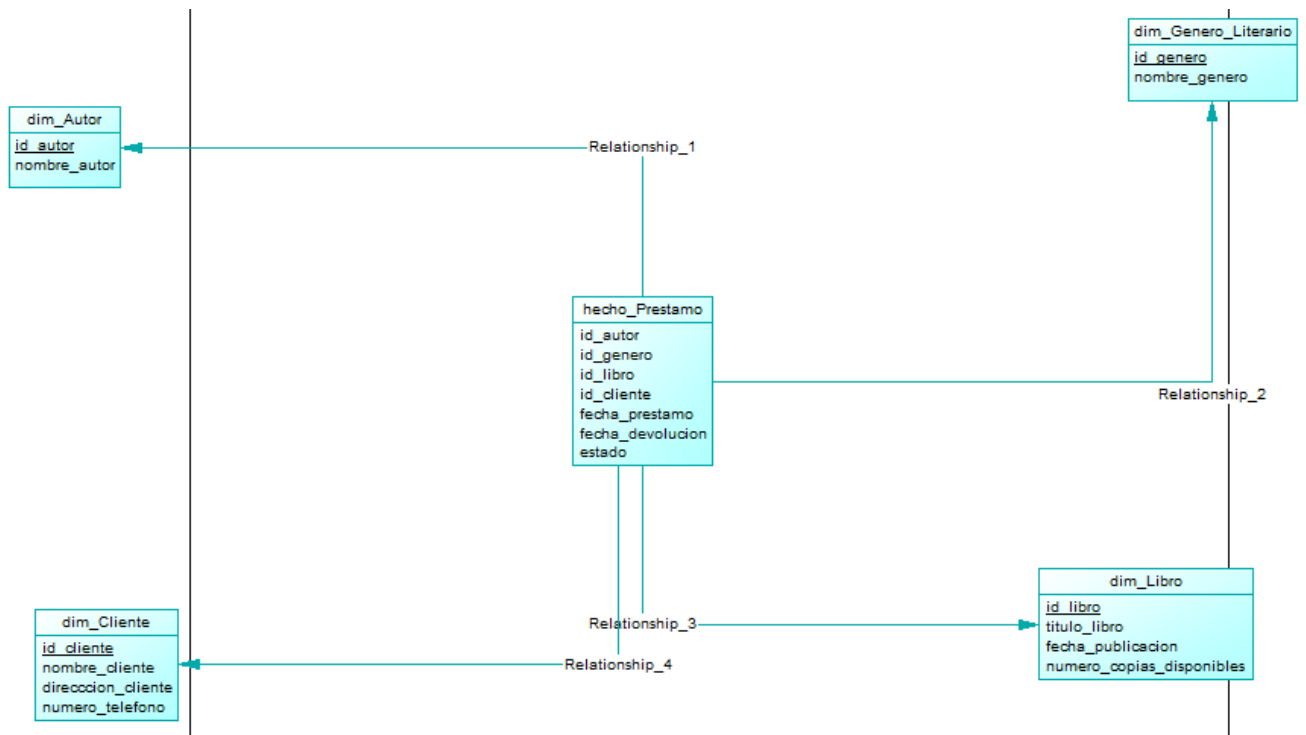
### ➤ Modelo Estrella - Modelo Conceptual




➤ **Modelo Estrella - Modelo Lógico**



➤ **Modelo Estrella - Modelo Físico**



	<b>DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN</b> <b>INFORME DE TRABAJO AUTÓNOMO</b>	<b>Unidad 1</b>	
		<b>N.º</b> <b>Informe</b> SII 202351 <u>001</u>	
		<b>Página:</b> 4 de .16.	

- **Tablas de Dimensiones**

- **DIM\_AUTOR**

**ID\_AUTOR:** Clave primaria única para la tabla DIM\_AUTOR.

**NOMBRE\_AUTOR:** Nombre del autor.

- **DIM\_CLIENTE**

**ID\_CLIENTE:** Clave primaria única para la tabla DIM\_CLIENTE.

**NOMBRE\_CLIENTE:** Nombre del cliente.

**DIRECCION\_CLIENTE:** Dirección del cliente.

**NUMERO\_TELEFONO:** Número de teléfono del cliente.

- **DIM\_GENERO\_LITERARIO**

**ID\_GENERO:** Clave primaria única para la tabla DIM\_GENERO\_LITERARIO.

**NOMBRE\_GENERO:** Nombre del género literario.

- **DIM\_LIBRO**

**ID\_LIBRO:** Clave primaria única para la tabla DIM\_LIBRO.

**TITULO\_LIBRO:** Título del libro.

**FECHA\_PUBLICACION:** Fecha de publicación del libro.

**NUMERO\_COPIAS\_DISPONIBLES:** Número de copias disponibles del libro.

- **Tabla Hecho:**

- **HECHOS\_PRESTAMO**

**ID\_AUTOR:** Clave foránea que referencia a la tabla DIM\_AUTOR.

**ID\_GENERO:** Clave foránea que referencia a la tabla DIM\_GENERO\_LITERARIO.


**ID\_LIBRO:** Clave foránea que referencia a la tabla DIM\_LIBRO.

**ID\_CLIENTE:** Clave foránea que referencia a la tabla DIM\_CLIENTE.

**FECHA\_PRESTAMO:** Fecha en que se realizó el préstamo.

**FECHA\_DEVOLUCION:** Fecha en que se devolvió el libro.

**ESTADO:** Estado actual del préstamo (por ejemplo, Pendiente, Devuelto, etc.).

	<p align="center"><b>DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN</b> <b>INFORME DE TRABAJO AUTÓNOMO</b></p>	<p align="center"><b>Unidad 1</b></p>
		<p>N.º Informe SII 202351 <u>001</u></p>
		<p>Página: 5 de .16.</p>

## ➤ Replicación en PostgreSQL

### Creación de la Red

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\chris> docker network create redExamen
dd77edd88913992ef1f42b679257114bfc43fb7282e7c2f1c2684f1ba49527af
PS C:\Users\chris>
```

### Contenedor del Server 1

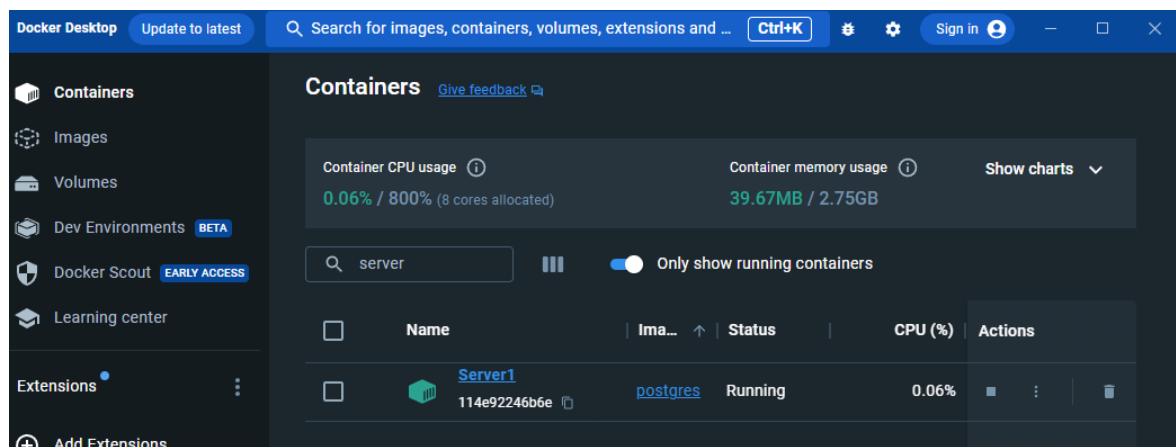
Creamos el contenedor tuneado cada uno de sus parámetros y asignando un volumen y la red creada.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.


Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

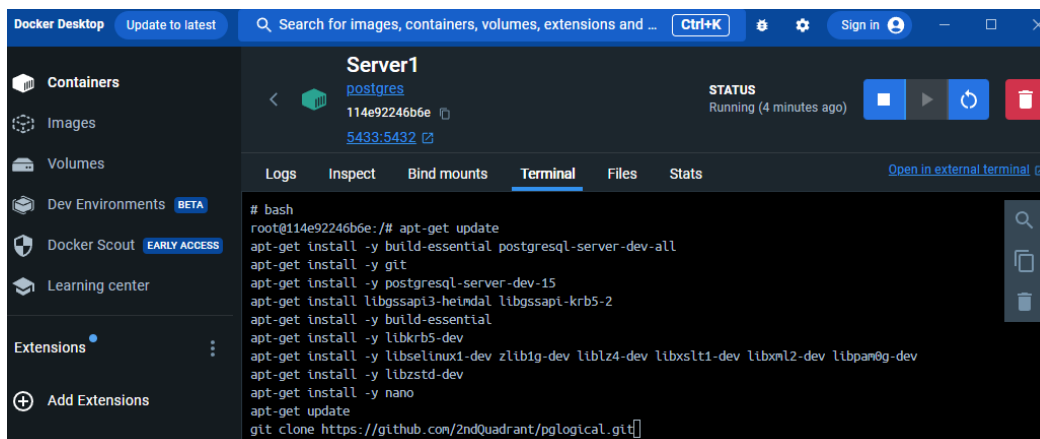
PS C:\Users\chris> docker run --name Server1 --network redExamen -e POSTGRES_PASSWORD=123456 -p 5433:5432 --cpus=1 --memory=500MB -d -v C:\Users\chris\Desktop\ExamenPostgres:/data/db --ulimit nofile=1024:1024 postgres
114e92246b6ef68bec6895a288ea9b0d13634b74a17d3a84ab86dab058c327f1
PS C:\Users\chris>
```

Verificamos en Docker la creación del contenedor.



Ingresamos al bash del contenedor y asignaremos los siguientes complementos necesarios para la replicación.

	<b>DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN</b> <b>INFORME DE TRABAJO AUTÓNOMO</b>	Unidad 1
		N.º Informe SII 202351 001
		Página: 6 de .16.



```
# bash
root@114e92246b6e:/# apt-get update
apt-get install -y build-essential postgresql-server-dev-all
apt-get install -y git
apt-get install -y postgresql-server-dev-15
apt-get install libgssapi3-heimdal libgssapi-krb5-2
apt-get install -y build-essential
apt-get install -y libkrb5-dev
apt-get install -y libsasl2-dev zlib1g-dev liblz4-dev libxslt1-dev libxml2-dev libpq-dev
apt-get install -y libzstd-dev
apt-get install -y nano
apt-get update
git clone https://github.com/2ndQuadrant/pglogical.git
```

Una vez terminado de instalarse los paquetes, ingresaremos a cd pglogical e instalaremos ahora el siguiente complemento.

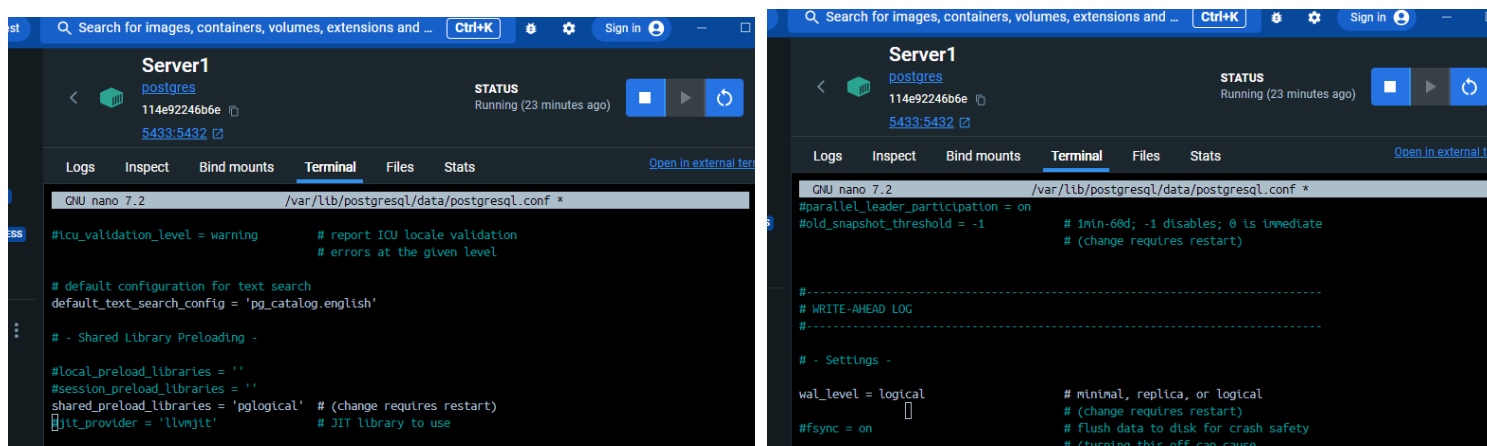
```
root@114e92246b6e:/# cd pglogical
root@114e92246b6e:/pglogical# apt-get install -y libxslt1-dev
```

```
root@114e92246b6e:/pglogical# apt-get install -y libpq-dev
```

```
root@114e92246b6e:/pglogical# make && make install
```

Ahora ingresaremos al editor de texto con el comando de nano y cambiaremos los siguientes parámetros los cuales *son* `shared_preload_libraries = 'pglogical'` y `wal_level = logical` lo podremos buscar presionando Ctrl + W. Luego de ello guardaremos los cambios con Ctrl + O y salimos con Ctrl + X.

```
nano /var/lib/postgresql/data/postgresql.conf
shared_preload_libraries = 'pglogical'
wal_level = logical
```





Reiniciamos nuestro contenedor para ejecutar los cambios.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\chris> docker stop server1
server1
PS C:\Users\chris> docker start server1
server1
PS C:\Users\chris>
```

Una vez realizado las configuraciones previas, ingresaremos al contenedor de la siguiente forma:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\chris> docker exec -it Server1 bash
root@114e92246b6e:/# psql -h localhost -p 5432 -U postgres
psql (16.1 (Debian 16.1-1.pgdg120+1))
Type "help" for help.

postgres=#
```


Creamos nuestra base de datos e ingresamos en ella.

```
postgres=# CREATE DATABASE libreria;
CREATE DATABASE
postgres=# \c libreria;
You are now connected to database "libreria" as user "postgres".
libreria=#
```

Creamos las tablas que previamente modelamos.

```
libreria=# create table DIM_AUTOR (
  ID_AUTOR          SERIAL              not null,
  NOMBRE_AUTOR      VARCHAR(30)         null,
  constraint PK_DIM_AUTOR primary key (ID_AUTOR)
);
CREATE TABLE
libreria=# create table DIM_CLIENTE (
  ID_CLIENTE        INT4                not null,
  NOMBRE_CLIENTE    VARCHAR(50)         null,
  DIRECCION_CLIENTE VARCHAR(20)         null,
  NUMERO_TELEFONO   VARCHAR(11)         null,
  constraint PK_DIM_CLIENTE primary key (ID_CLIENTE)
);
CREATE TABLE
libreria=# create table DIM_GENERO_LITERARIO (
  ID_GENERO          SERIAL              not null,
  NOMBRE_GENERO      CHAR(30)            null,
  constraint PK_DIM_GENERO_LITERARIO primary key (ID_GENERO)
);
CREATE TABLE
libreria=# create table DIM_LIBRO (
  ID_LIBRO          INT4                not null,
  TITULO_LIBRO      CHAR(30)            null,
  FECHA_PUBLICACION DATE                null,
  NUMERO_COPIAS_DISPONIBLES INT4        null,
  constraint PK_DIM_LIBRO primary key (ID_LIBRO)
);
CREATE TABLE
```

```
libreria=# create table HECHO_PRESTAMO (
  ID_AUTOR          INT4                null,
  ID_GENERO          INT4                null,
  ID_LIBRO           INT4                null,
  ID_CLIENTE         INT4                null,
  FECHA_PRESTAMO     DATE                null,
  FECHA_DEVOLUCION   DATE                null,
  ESTADO             VARCHAR(15)        null
);
CREATE TABLE
libreria=#
```

	<b>DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN</b> <b>INFORME DE TRABAJO AUTÓNOMO</b>	<b>Unidad 1</b>	
		<b>N.º</b>	<b>SII 202351 001</b>
		<b>Página:</b>	8 de .16.

Creamos una extensión

```
CREATE EXTENSION pglogical
```

Aplicamos la replicación

```
SELECT pg_create_logical_replication_slot('slot_1', 'pgoutput');
```

Creamos un nodo con los siguientes parámetros

```
SELECT pglogical.create_node(node_name := 'node1', dsn :=
'host=localhost port=5432 dbname=examen user=postgres
password=123456');
```

Aplicamos un set a la replicación

```
SELECT pglogical.create_replication_set('replication_set_1', true);
```

Añadimos las tablas que creamos anteriormente para la replicación

```
SELECT pglogical.replication_set_add_table('replication_set_1',
'public.DIM_AUTOR', true);
SELECT pglogical.replication_set_add_table('replication_set_1',
'public.DIM_CLIENTE', true);
SELECT pglogical.replication_set_add_table('replication_set_1',
'public.DIM_GENERO_LITERARIO', true);
SELECT pglogical.replication_set_add_table('replication_set_1',
'public.DIM_LIBRO', true);
SELECT pglogical.replication_set_add_table('replication_set_1',
'public.HECHO_PRESTAMO', true);
```

Agregamos una clave primaria en nuestra tabla de hecho\_prestamo.

```
ALTER TABLE HECHO_PRESTAMO ADD COLUMN ID_PRESTAMO
SERIAL PRIMARY KEY;
```



```
libreria=# CREATE EXTENSION pglogical;
CREATE EXTENSION
libreria=# SELECT pg_create_logical_replication_slot('slot_1', 'pgoutput');
pg_create_logical_replication_slot
-----
(slot_1,0/1A446D8)
(1 row)

libreria=# SELECT pglogical.create_node(node_name := 'node1', dsn := 'host=localhost port=5432 dbname=libreria user=postgres password=123456');
create_node
-----
1148549230
(1 row)

libreria=# SELECT pglogical.create_replication_set('replication_set_1', true);
create_replication_set
-----
242457877
(1 row)

libreria=# SELECT pglogical.replication_set_add_table('replication_set_1', 'public.DIM_AUTOR', true);
replication_set_add_table
-----
t
(1 row)

libreria=# SELECT pglogical.replication_set_add_table('replication_set_1', 'public.DIM_CLIENTE', true);
replication_set_add_table
-----
t
(1 row)


libreria=# SELECT pglogical.replication_set_add_table('replication_set_1', 'public.DIM_GENERO_LITERARIO', true);
replication_set_add_table
-----
t
(1 row)

libreria=# SELECT pglogical.replication_set_add_table('replication_set_1', 'public.DIM_LIBRO', true);
replication_set_add_table
-----
t
(1 row)
```

```
libreria=# ALTER TABLE HECHO_PRESTAMO ADD COLUMN ID_PRESTAMO SERIAL PRIMARY KEY;
ALTER TABLE
libreria=#
```

Salimos de pglogical con exit e ingresamos al root para identificar nuestro ip del contenedor del server1.

```
libreria=# exit
root@114e92246b6e:/# hostname -i
172.30.0.2
root@114e92246b6e:/#
```

	<b>DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN</b> <b>INFORME DE TRABAJO AUTÓNOMO</b>	Unidad 1	
		N.º Informe	SII 202351 001
		Página:	10 de .16.

## Contenedor del Server 2

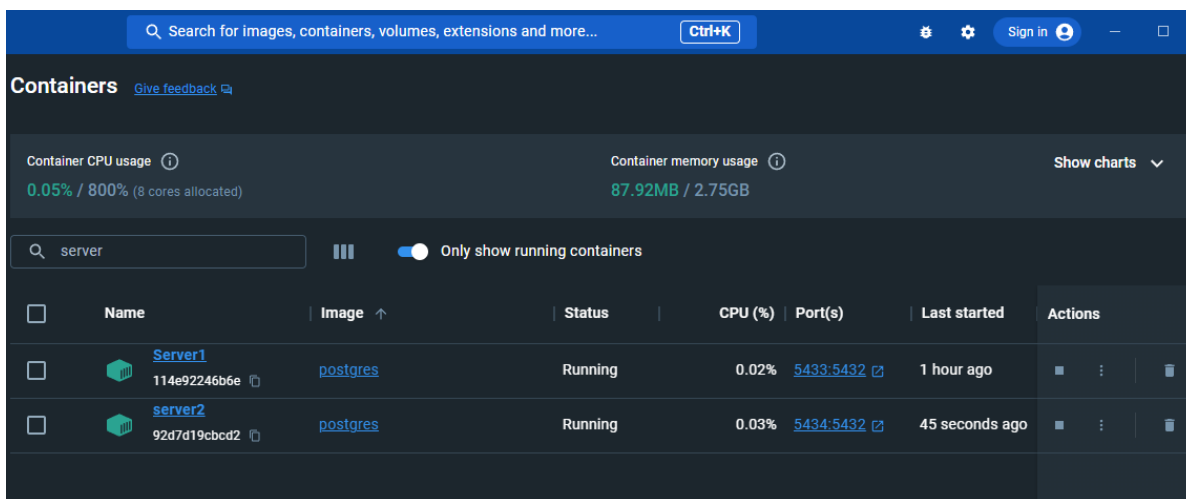
Creamos el contenedor tuneado cada uno de sus parámetros y asignando un volumen y la red creada.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\chris> docker run --name server2 --network redExamen -e POSTGRES_PASSWORD=123456 -p 5434:5432 --cpus=1 --memory=500MB -d -v C:\Users\chris\Desktop\ExamenPostgres2:/data/db --ulimit nofile=1024:1024 postgres
92d7d19cbcd2f5d41043cb2c26302f095544f17b04d4eb128456851b504ec5a7
PS C:\Users\chris>
```

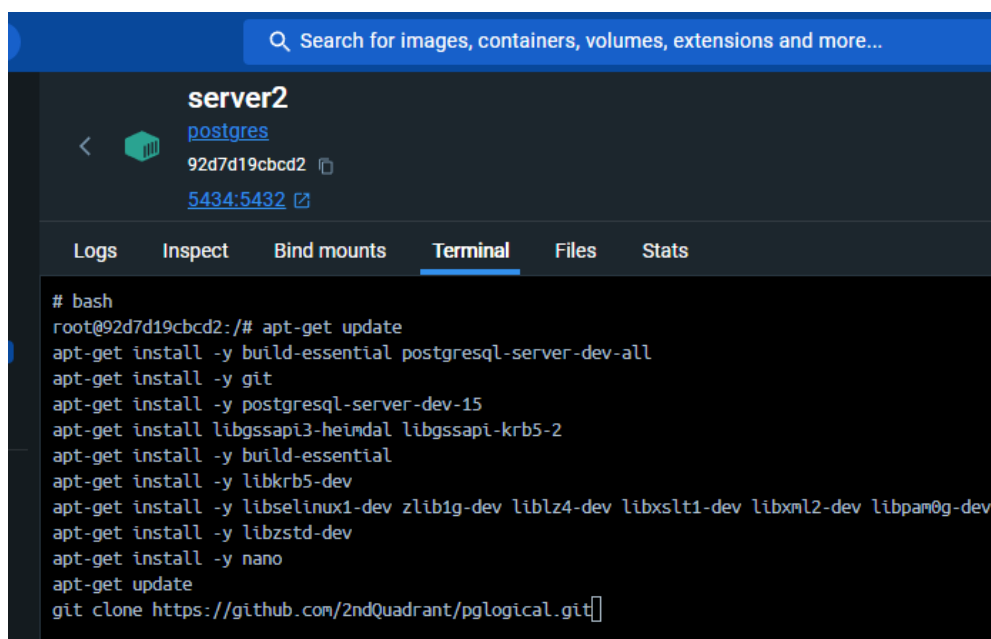
Verificamos en Docker la creación del contenedor.



The screenshot shows the Docker Desktop interface. At the top, there's a search bar and a 'Ctrl+K' button. Below that, the 'Containers' section is active, showing a summary of container usage: CPU usage at 0.05% / 800% (8 cores allocated) and memory usage at 87.92MB / 2.75GB. A table lists running containers:


Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
Server1 114e92246b6e	postgres	Running	0.02%	5433:5432	1 hour ago	[Stop] [Refresh] [Delete]
server2 92d7d19cbcd2	postgres	Running	0.03%	5434:5432	45 seconds ago	[Stop] [Refresh] [Delete]

Ingresamos al bash del contenedor y asignaremos los siguientes complementos necesarios para la replicación.



The screenshot shows the Docker Desktop interface with the 'Terminal' tab selected for the container 'server2'. The terminal output shows the following commands being executed:

```
# bash
root@92d7d19cbcd2:/# apt-get update
apt-get install -y build-essential postgresql-server-dev-all
apt-get install -y git
apt-get install -y postgresql-server-dev-15
apt-get install libgssapi3-heimdal libgssapi-krb5-2
apt-get install -y build-essential
apt-get install -y libkrb5-dev
apt-get install -y libselinux1-dev zlib1g-dev liblz4-dev libxslt1-dev libxml2-dev libpam0g-dev
apt-get install -y libzstd-dev
apt-get install -y nano
apt-get update
git clone https://github.com/2ndQuadrant/pglogical.git
```

 <b>ESPE</b> UNIVERSIDAD DE LAS FUERZAS ARMADAS INNOVACIÓN PARA LA EXCELENCIA	<b>DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN</b> <b>INFORME DE TRABAJO AUTÓNOMO</b>	Unidad 1
		N.º Informe SII 202351 <u>001</u>
		Página: 11 de .16.

Una vez terminado de instalarse los paquetes, ingresaremos a *cd pglogical* e instalaremos ahora los siguientes complementos.

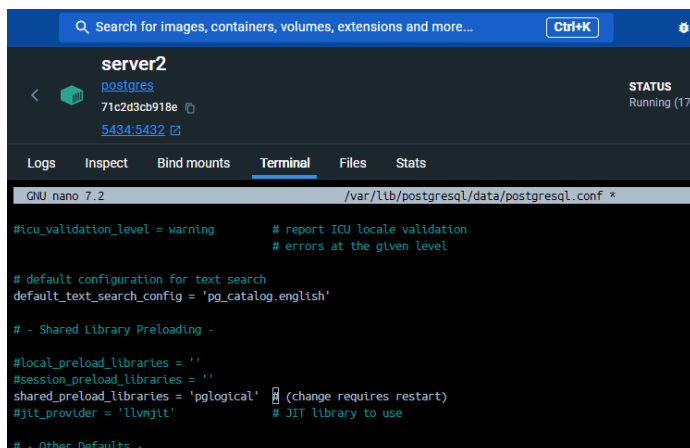
```
Resolving deltas: 100% (4117/4117), done.
root@71c2d3cb918e:/# cd pglogical
root@71c2d3cb918e:/pglogical# apt-get install -y libxslt1-dev
```

```
root@71c2d3cb918e:/pglogical# apt-get install -y libpam0g-dev
```

```
root@71c2d3cb918e:/pglogical# make && make install
```

Ahora ingresaremos al editor de texto con el comando de nano y cambiaremos los siguientes parámetros los cuales son `shared_preload_libraries = 'pglogical'` y `wal_level = logical` lo podremos buscar presionando Ctrl + W. Luego de ello guardaremos los cambios con Ctrl + O y salimos con Ctrl + X.

```
root@71c2d3cb918e:/pglogical# nano /var/lib/postgresql/data/postgresql.conf
```



```
server2
postgres
71c2d3cb918e
5434:5432

Logs Inspect Bind mounts Terminal Files Stats

GNU nano 7.2 /var/lib/postgresql/data/postgresql.conf *

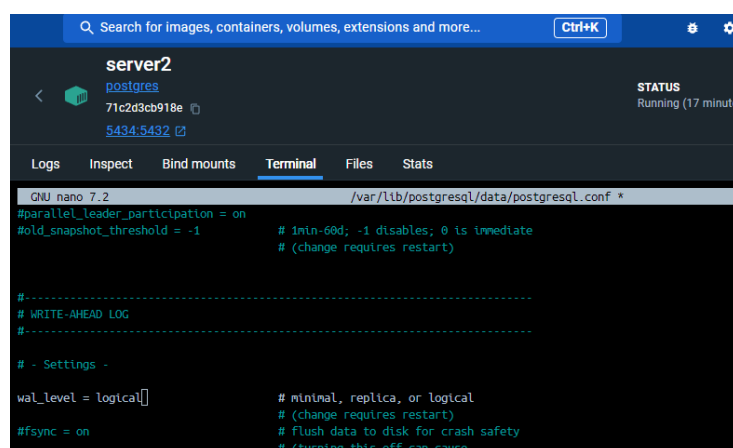
#icu_validation_level = warning          # report ICU locale validation
#                                         # errors at the given level

# default configuration for text search
default_text_search_config = 'pg_catalog.english'

# - Shared Library Preloading -

#local_preload_libraries = ''
#session_preload_libraries = ''
shared_preload_libraries = 'pglogical'  (change requires restart)
#jit_provider = 'llvmlite'              # JIT library to use

# - Other Defaults -
```



```
server2
postgres
71c2d3cb918e
5434:5432

Logs Inspect Bind mounts Terminal Files Stats

GNU nano 7.2 /var/lib/postgresql/data/postgresql.conf *

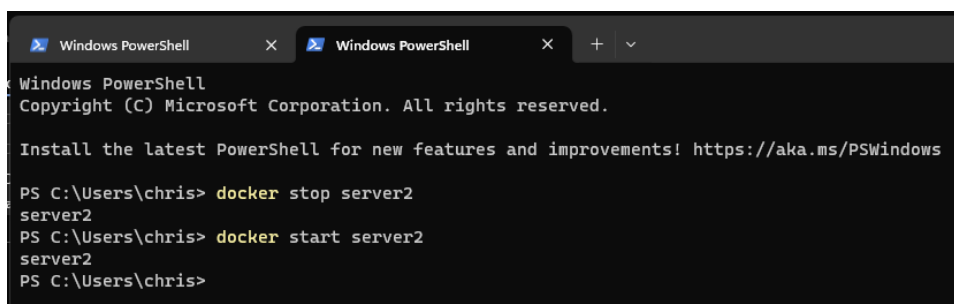
#parallel_leader_participation = on
#old_snapshot_threshold = -1           # 1min-60d; -1 disables; 0 is immediate
#                                         # (change requires restart)

#-----
# WRITE-AHEAD LOG
#-----

# - Settings -

wal_level = logical                    # minimal, replica, or logical
#                                         # (change requires restart)
#flush = on                            # Flush data to disk for crash safety
#                                         # (turning this off can cause
```


Reiniciamos nuestro contendedor para ejecutar los cambios.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\chris> docker stop server2
server2
PS C:\Users\chris> docker start server2
server2
PS C:\Users\chris>
```

	DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN INFORME DE TRABAJO AUTÓNOMO	Unidad 1	
		N.º	SII 202351 <u>001</u>
		Informe	Página: 12 de .16.

Una vez realizado las configuraciones previas, ingresaremos al contenedor de la siguiente forma:

```
PS C:\Users\chris> docker exec -it server2 bash
root@71c2d3cb918e:/# psql -h localhost -p 5432 -U postgres
psql (16.1 (Debian 16.1-1.pgdg120+1))
Type "help" for help.


postgres=#
```

Creamos nuestra base de datos e ingresamos en ella.

```
postgres=# CREATE DATABASE libreria;
CREATE DATABASE
postgres=# \c libreria;
You are now connected to database "libreria" as user "postgres".
libreria=#
```

Creamos las tablas que previamente modelamos.

```
libreria=# create table DIM_AUTOR (
  ID_AUTOR          SERIAL          not null,
  NOMBRE_AUTOR      VARCHAR(30)     null,
  constraint PK_DIM_AUTOR primary key (ID_AUTOR)
);
CREATE TABLE
libreria=# create table DIM_CLIENTE (
  ID_CLIENTE        INT4            not null,
  NOMBRE_CLIENTE    VARCHAR(50)     null,
  DIRECCION_CLIENTE VARCHAR(20)     null,
  NUMERO_TELEFONO   VARCHAR(11)     null,
  constraint PK_DIM_CLIENTE primary key (ID_CLIENTE)
);
CREATE TABLE
libreria=# create table DIM_GENERO_LITERARIO (
  ID_GENERO         SERIAL          not null,
  NOMBRE_GENERO     CHAR(30)        null,
  constraint PK_DIM_GENERO_LITERARIO primary key (ID_GENERO)
);
CREATE TABLE
libreria=# create table DIM_LIBRO (
  ID_LIBRO          INT4            not null,
  TITULO_LIBRO      CHAR(30)        null,
  FECHA_PUBLICACION DATE            null,
  NUMERO_COPIAS_DISPONIBLES INT4    null,
  constraint PK_DIM_LIBRO primary key (ID_LIBRO)
);
CREATE TABLE
libreria=# create table HECHO_PRESTAMO (
  ID_AUTOR          INT4            null,
  ID_GENERO         INT4            null,
  ID_LIBRO          INT4            null,
  ID_CLIENTE        INT4            null,
  FECHA_PRESTAMO    DATE            null,
  FECHA_DEVOLUCION  DATE            null,
  ESTADO            VARCHAR(15)     null
);
CREATE TABLE
libreria=#
```

	<b>DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN</b> <b>INFORME DE TRABAJO AUTÓNOMO</b>	<b>Unidad 1</b>
		<b>N.º</b> <b>Informe</b> SII 202351 <u>001</u>
		<b>Página:</b> 13 de .16.

Creamos una extensión

```
CREATE EXTENSION pglogical
```

Creamos un nodo con los siguientes parámetros

```
SELECT pglogical.create_node(node_name := 'node2', dsn :=
'host=localhost port=5432 dbname=libreria user=postgres
password=123456');
```

Aplicamos un slot a la replicación

```
SELECT slot_name FROM pg_replication_slots WHERE plugin =
'pglogical_output';
```

Creamos la suscripción teniendo en cuenta la dirección ip del contenedor Server1

```
SELECT pglogical.create_subscription(
subscription_name := 'subscription_1',
provider_dsn := 'host=172.30.0.2 port=5432 dbname=libreria
user=postgres password=123456'
);
```

Realizamos un replica set de la siguiente forma:


```
SELECT pglogical.create_replication_set('replication_set_1', true);
```

Configuramos la tablas que queremos replicar en nuestro set.

```
SELECT pglogical.replication_set_add_table('replication_set_1', 'public.DIM_AUTOR',
true);
SELECT pglogical.replication_set_add_table('replication_set_1', 'public.DIM_CLIENTE',
true);
SELECT pglogical.replication_set_add_table('replication_set_1',
'public.DIM_GENERO_LITERARIO', true);
SELECT pglogical.replication_set_add_table('replication_set_1', 'public.DIM_LIBRO',
true);
SELECT pglogical.replication_set_add_table('replication_set_1',
'public.HECHO_PRESTAMO', true);
```

Agregamos una clave primaria en nuestra tabla de hecho\_prestamo.

```
ALTER TABLE HECHO_PRESTAMO ADD COLUMN ID_PRESTAMO
SERIAL PRIMARY KEY;
```

	DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN INFORME DE TRABAJO AUTÓNOMO	Unidad 1	
		N.º	SII 202351 <u>001</u>
		Informe	Página: 14 de .16.

Configuramos nuestro set de replicación a la suscripción.

```
SELECT
pglogical.alter_subscription_add_replication_set('subscription_1',
'replication_set_1');
```

```
libreria=# CREATE EXTENSION pglogical;
CREATE EXTENSION
libreria=# SELECT pglogical.create_node(node_name := 'node2', dsn := 'host=localhost port=5432 dbname=libreria user=postgres password=123456');
create_node
-----
3367056606
(1 row)

libreria=# SELECT slot_name FROM pg_replication_slots WHERE plugin = 'pglogical_output';
slot_name
-----
(0 rows)
```

```
libreria=# SELECT pglogical.create_subscription(
subscription_name := 'subscription_1',
provider_dsn := 'host=172.30.0.2 port=5432 dbname=libreria user=postgres password=123456'
);
create_subscription
-----
1283614838
(1 row)

libreria=# SELECT pglogical.create_replication_set('replication_set_1', true);
create_replication_set
-----
3308670300
(1 row)

libreria=# ALTER TABLE HECHO_PRESTAMO ADD COLUMN ID_PRESTAMO SERIAL PRIMARY KEY;
ALTER TABLE
libreria=# SELECT pglogical.alter_subscription_add_replication_set('subscription_1', 'replication_set_1');
alter_subscription_add_replication_set
-----
t
(1 row)

libreria=#
```

Salimos de pglogical con exit e ingresamos al root para identificar nuestro ip del contenedor del server2.

```
libreria=# exit
root@71c2d3cb918e:/# hostname -i
172.30.0.3
root@71c2d3cb918e:/#
```

Regresamos al server1 para configurar nuestro set de replicación a la suscripción.

```
SELECT pglogical.create_subscription(
    subscription_name := 'subscription_1',
    provider_dsn := 'host=172.30.0.3 port=5432 dbname=libreria
    user=postgres password=123456'
);

SELECT
pglogical.alter_subscription_add_replication_set('subscription_1',
'replication_set_1');
```

```
postgres=# exit
root@114e92246b6e:/# hostname -i
172.30.0.2
root@114e92246b6e:/# psql -h localhost -p 5432 -U postgres
psql (16.1 (Debian 16.1-1.pgdg120+1))
Type "help" for help.

postgres=# \c libreria;
You are now connected to database "libreria" as user "postgres".
libreria=# SELECT pglogical.create_subscription(
    subscription_name := 'subscription_1',
    provider_dsn := 'host=172.30.0.3 port=5432 dbname=libreria user=postgres password=123456'
);
 create_subscription
-----
1283614838
(1 row)

libreria=# SELECT pglogical.alter_subscription_add_replication_set('subscription_1', 'replication_set_1');
 alter_subscription_add_replication_set
-----
t
(1 row)

libreria=#
```

Verificamos la replicación haciendo inserciones en el Server1.

Realizaremos una inserción en la tabla de DIM\_AUTOR.


```
libreria=# INSERT INTO DIM_AUTOR (NOMBRE_AUTOR) VALUES
('Gabriel García Márquez'),
('J.K. Rowling'),
('Isabel Allende');
INSERT 0 3
```

Verificaremos si la inserción mediante una consulta en el server2.

```
libreria=# SELECT * FROM DIM_AUTOR;
 id_autor | nombre_autor
-----+-----
1 | Gabriel García Márquez
2 | J.K. Rowling
3 | Isabel Allende
(3 rows)

libreria=#
```



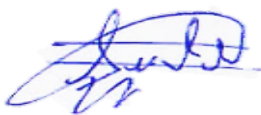
	<b>DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN</b> <b>INFORME DE TRABAJO AUTÓNOMO</b>	Unidad 1	
		N.º	SII 202351 <u>001</u>
		Página:	16 de .16.

## F. GITHUB

Enlace del Repositorio:

<https://github.com/ChristianAndrango/Examen--Postgres>

## G. NOMBRES, APELLIDOS Y FIRMA DEL ESTUDIANTE



Christian Jhonatan Andrango Carchipulla

## G. FECHA DE ENTREGA

Sangolquí, 13 de diciembre del 2023