

UNIVERSIDAD DE LAS AMERICAS

FACULTAD DE INGENIERIAS Y CIENCIAS APLICADAS

PROYECTO INTEGRADOR

ADMIN DE BASES DE DATOS

**IMPLEMENTACION DE UNA SOLUCION DE BASES DE DATOS CONTENERIZADA**

JEAN PIERRE MOYANO, JUAN CALVACHI, ARIEL CARRILLO

2024-10

1. **DESCRIPCION DE LA SOLUCION IMPLEMENTADA**

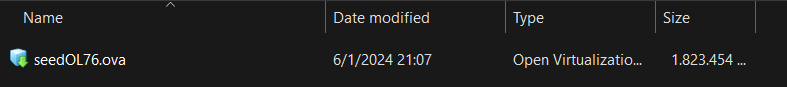
La solución implementada consiste en el despliegue de una base de datos contenerizada (contenedor de postgres) en una máquina virtual de con SO Oracle-Linux de 64 bits, que permitirá almacenar y gestionar los datos para una empresa de transporte y recorridos escolares.

Se aplican las prácticas óptimas recomendadas para garantizar que los archivos de configuración y los datos de la base de datos cuenten con persistencia.

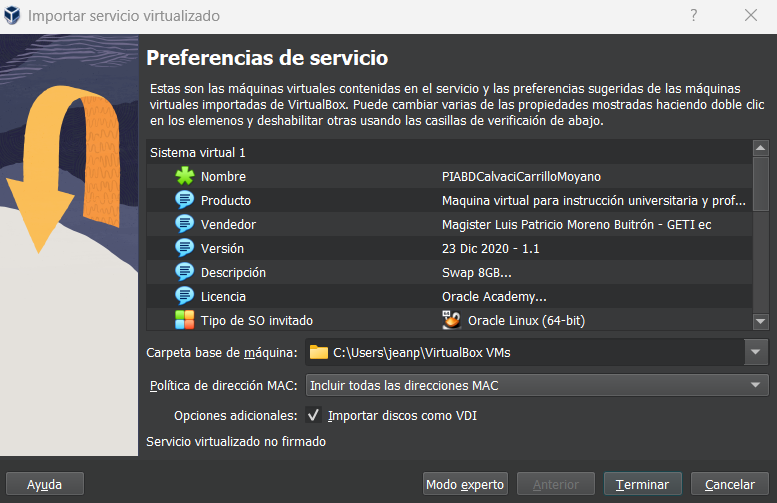
1. **DESARROLLO**
2. Despliegue de la solución de bases de datos contenerizada

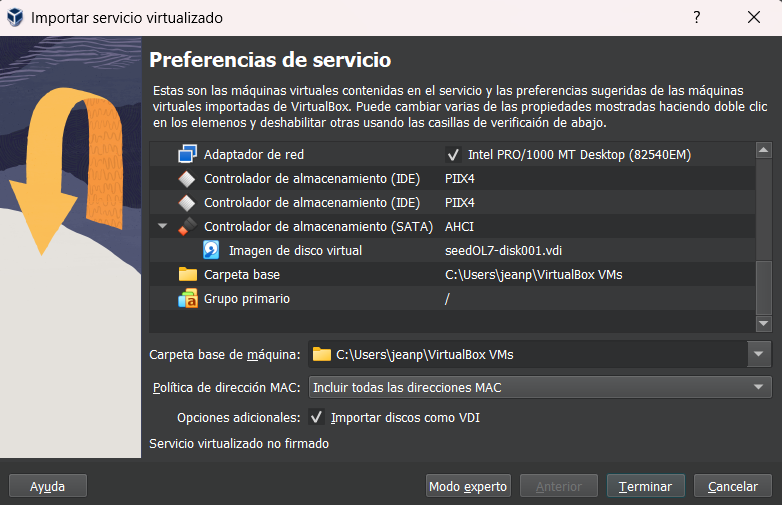
Instalación:

Para implementar la solución requerida es necesario configurar una máquina virtual, en nuestro caso Oracle Linux, se adjunta una imagen del archivo .ova a utilizar:

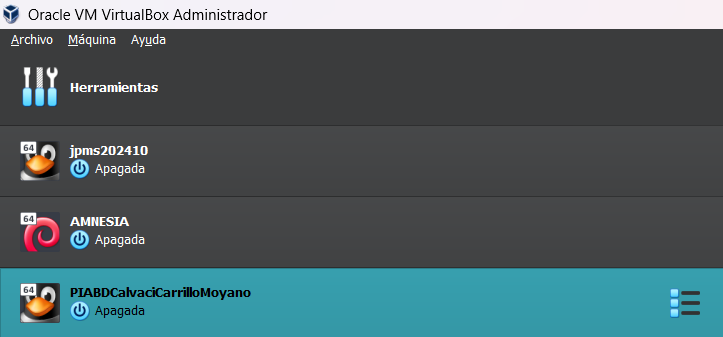


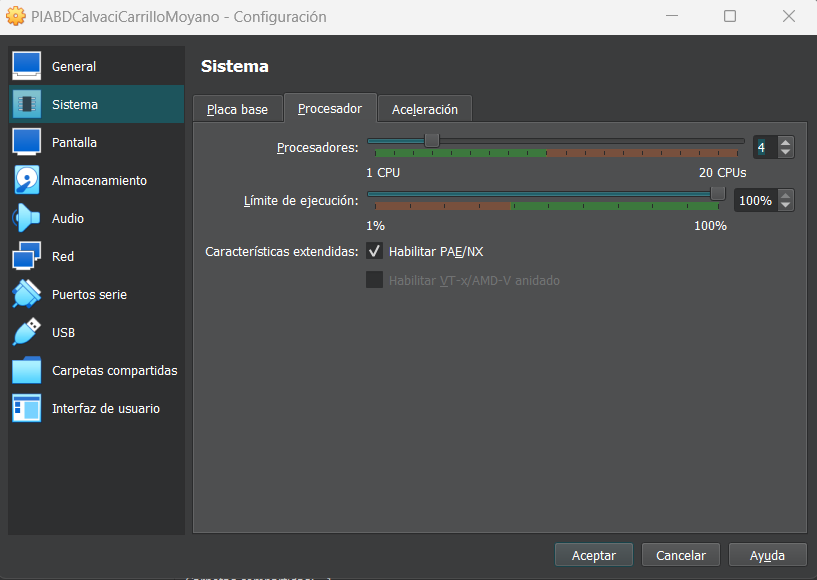
Se ha de importar el recurso virtualizado, realizado con la siguiente configuración:

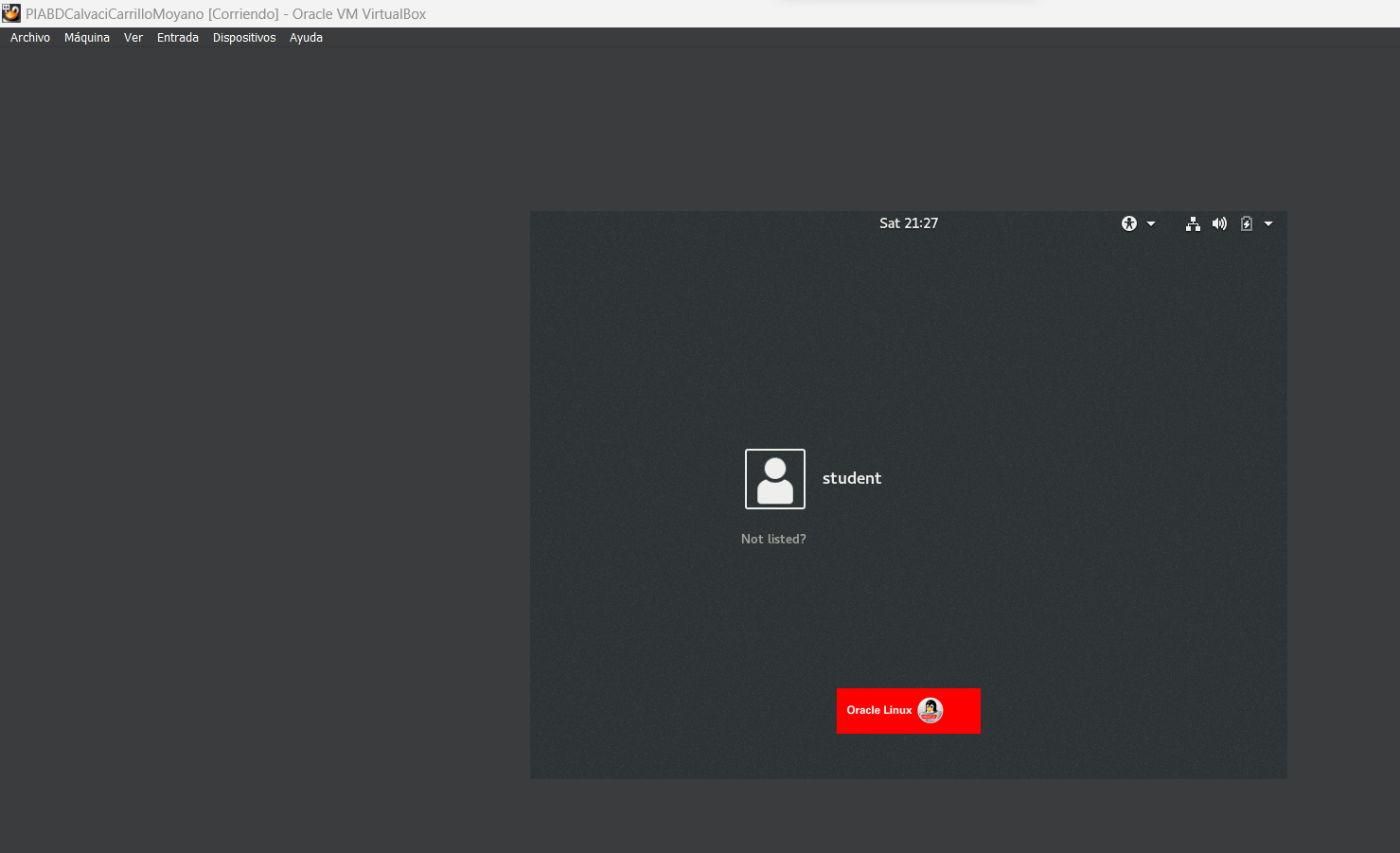




Tras realizar validar que el import se realizó correctamente fue necesario realizar modificaciones sobre las configuraciones de la máquina virtual.

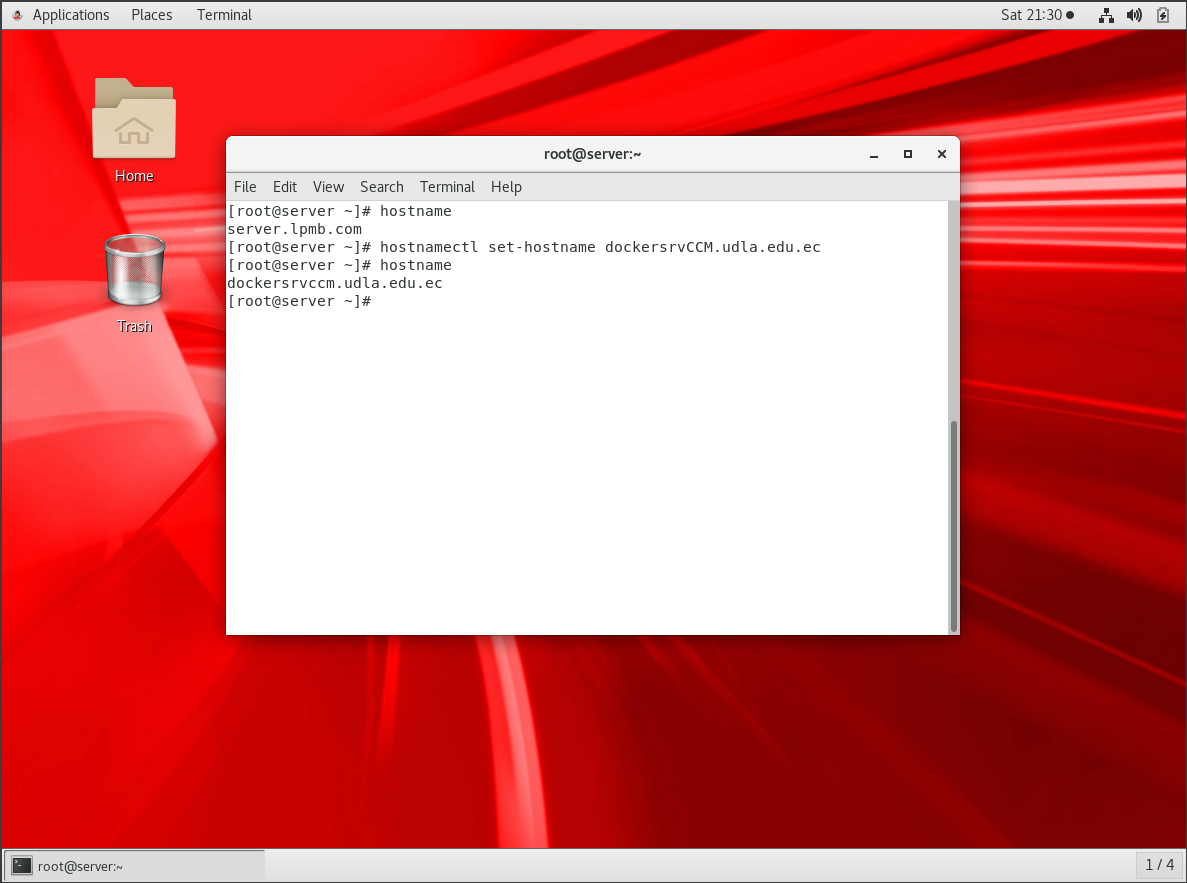


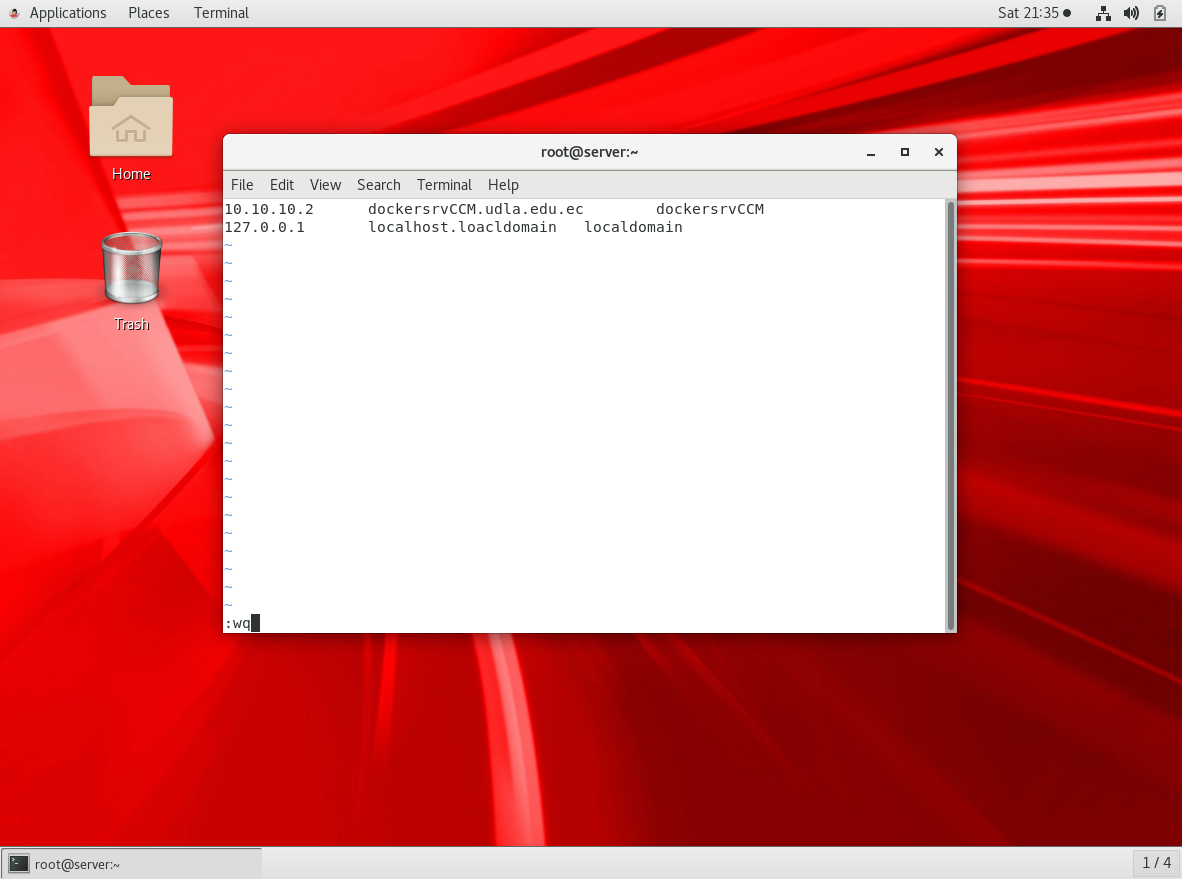




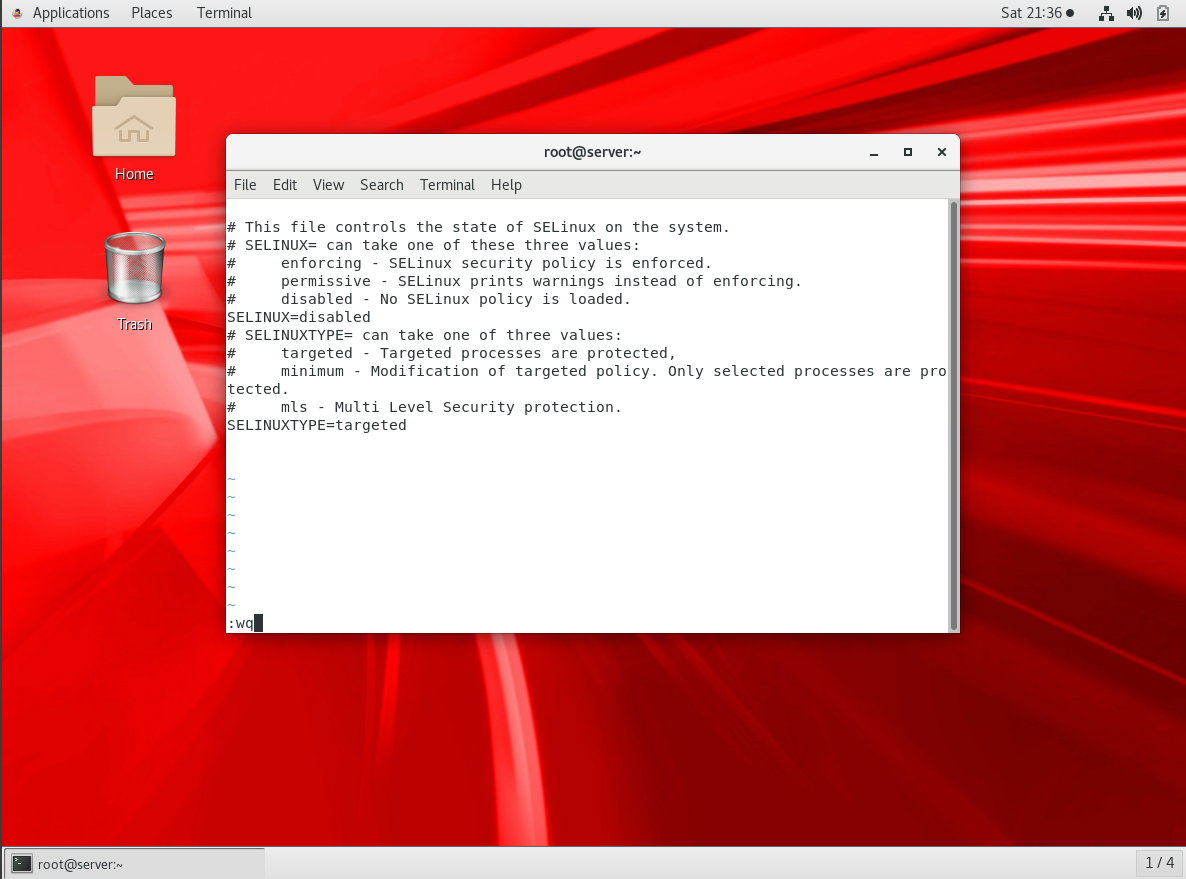
Tras tener acceso a la máquina, ingresamos al usuario root y dentro de este realizamos siguientes modificaciones:

1. Configuración de hostname.

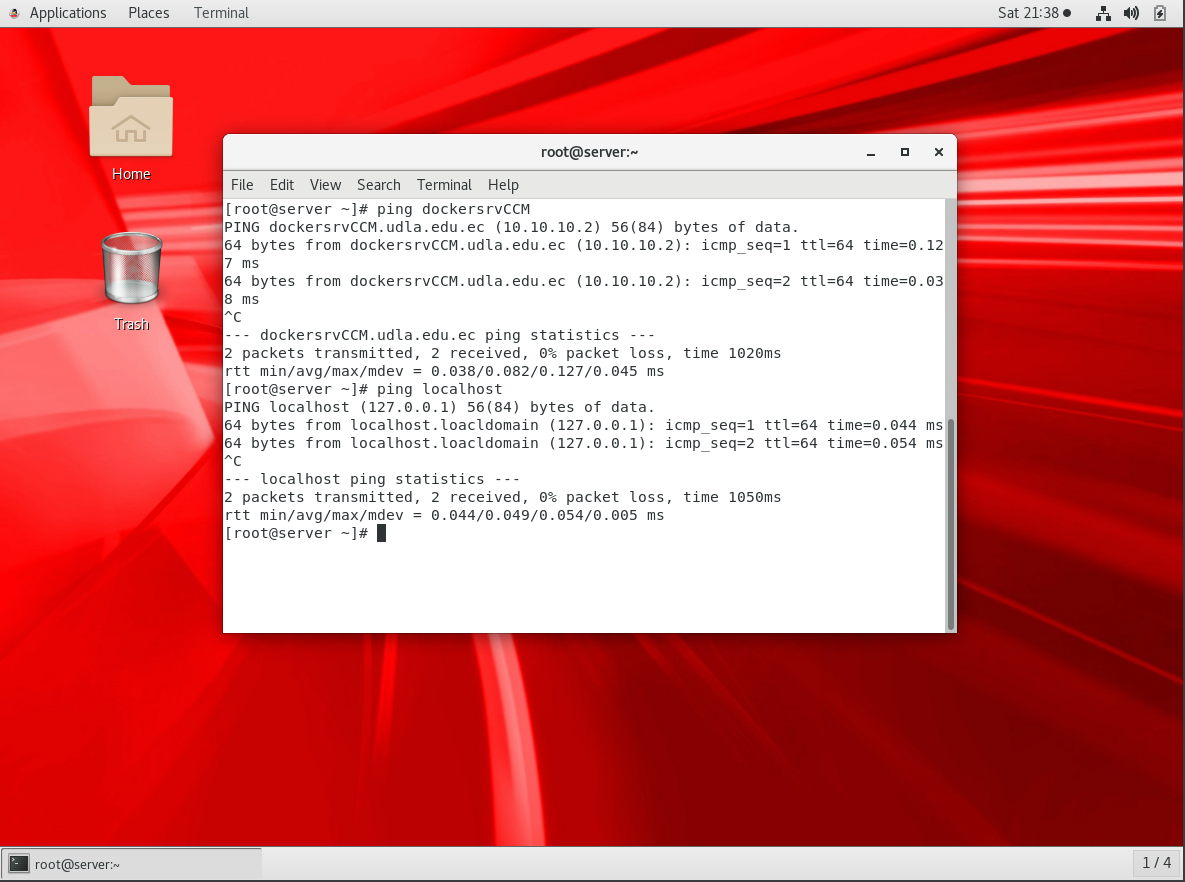




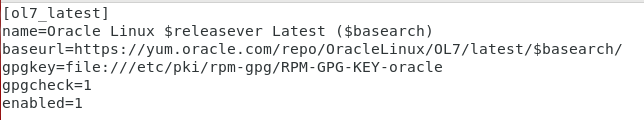
1. Configuración de SELinux: Se colocó su estado como “disabled” para evitar problemas en función de las recomendaciones del docente.

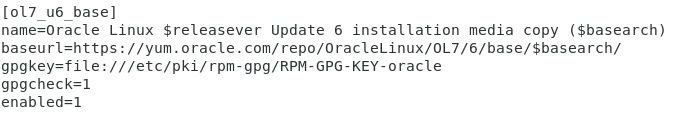


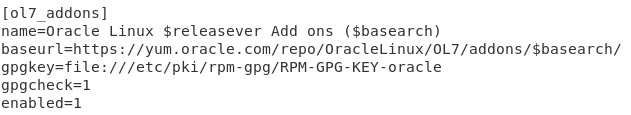
1. Pruebas de conectividad mediante ping para validar configuraciones previas.



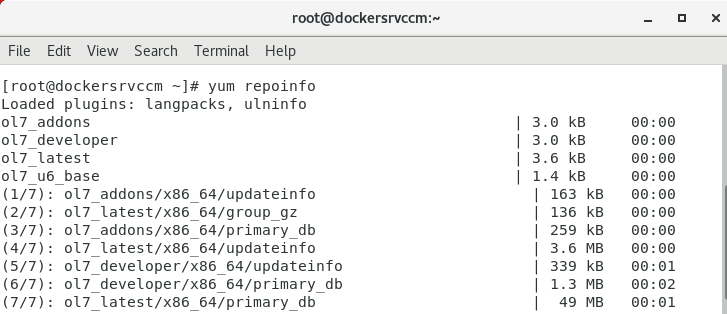
1. Modificar el archivo public-yum-ol7.repo



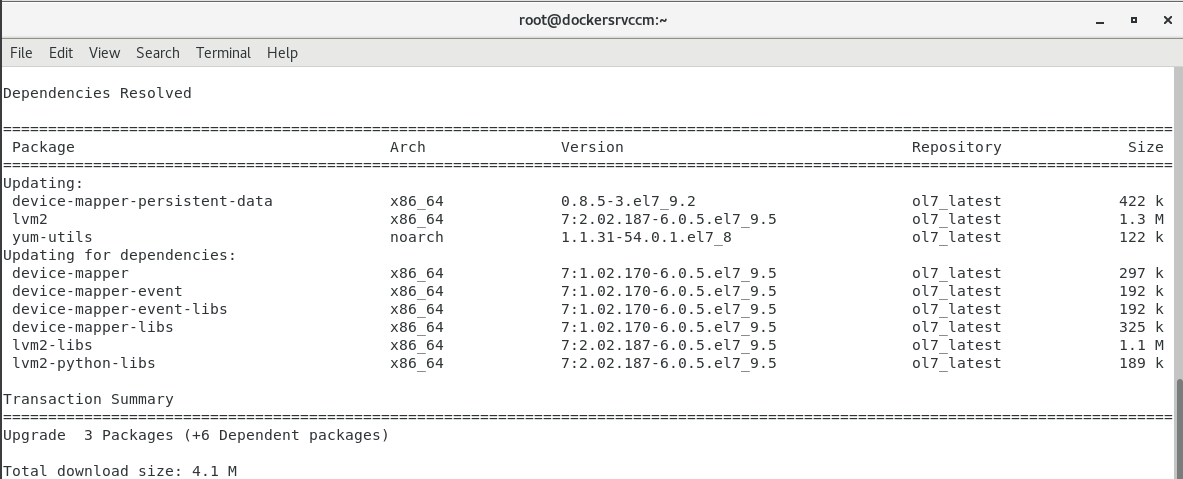


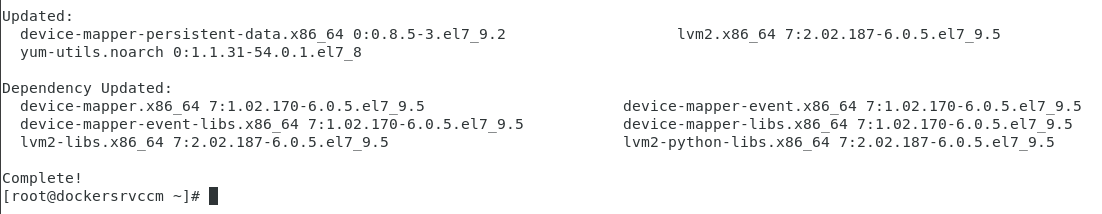




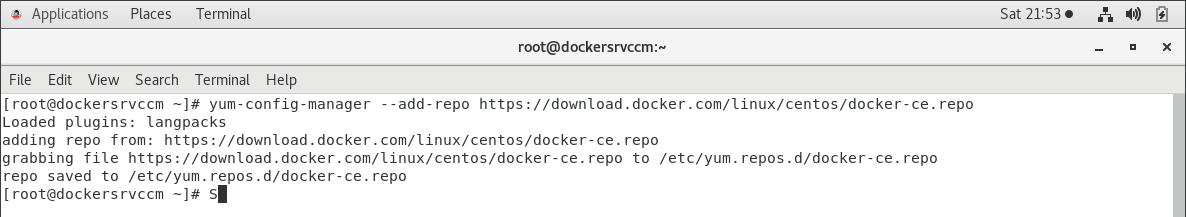


1. Instalación de paquetes





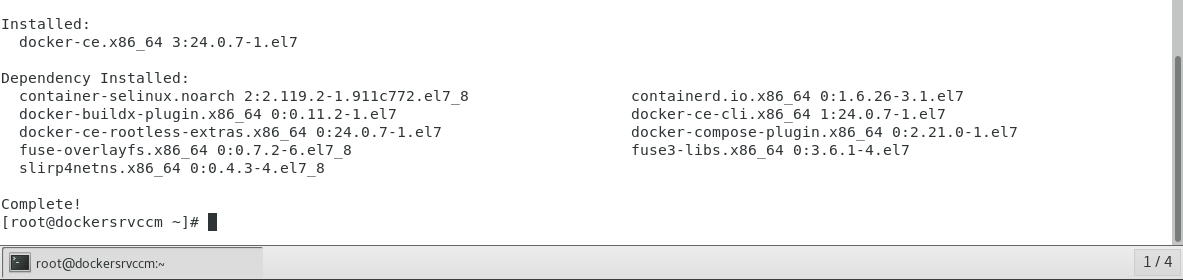
1. Agregar el repositorio de Docker CE a la lista de repositorios conocidos por YUM en un CentOS.



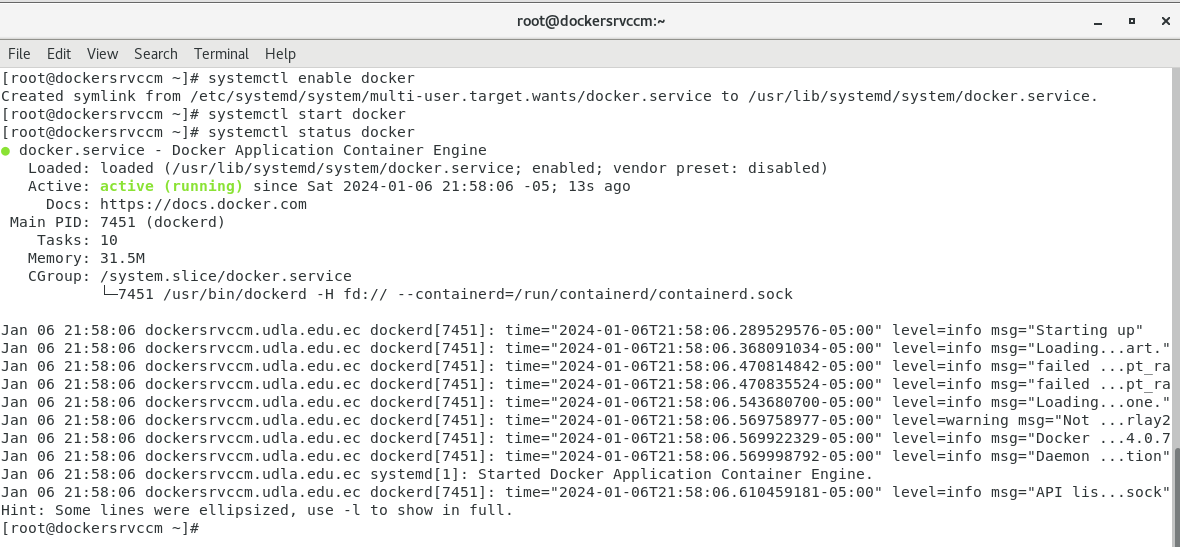


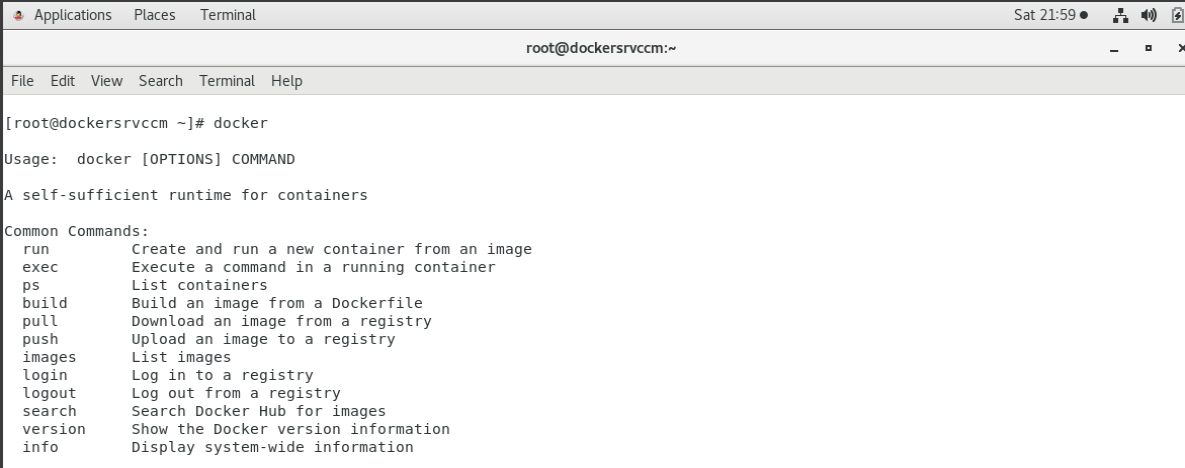
1. Instalar Docker



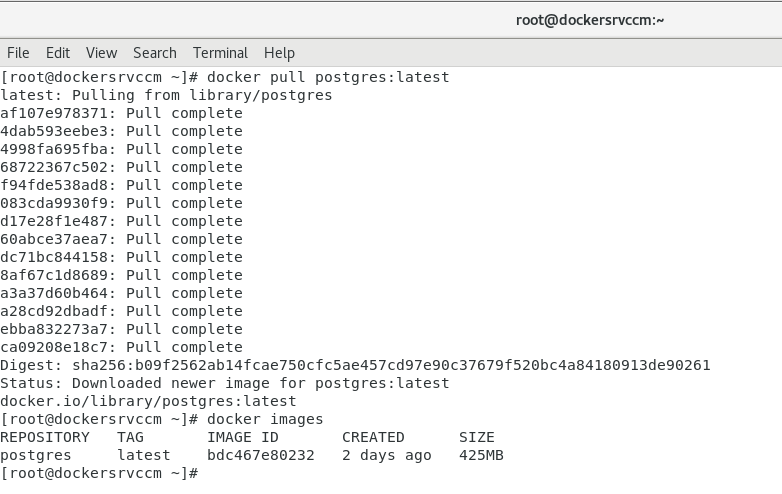


1. Configurar el comportamiento de Docker Engine como daemon y verificar su funcionamiento.

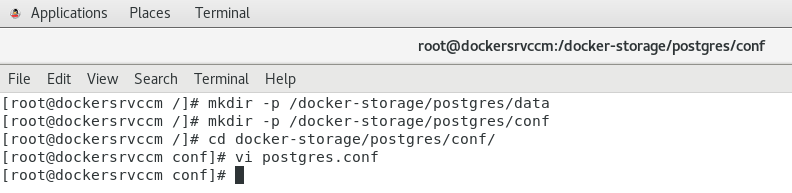




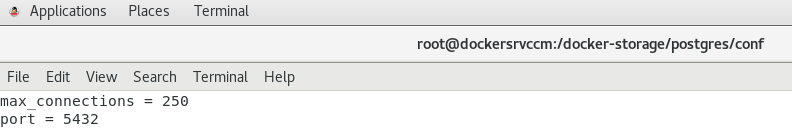
1. Descargar la imagen de postgres

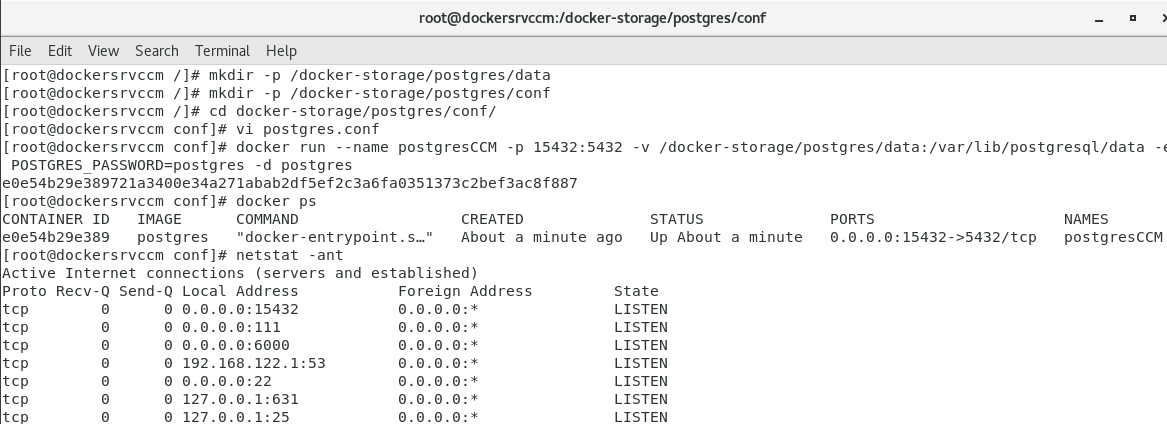


1. Archivos de configuración en almacenamiento persistente
2. Crear directorios para persistencia de datos y archivo de configuración.

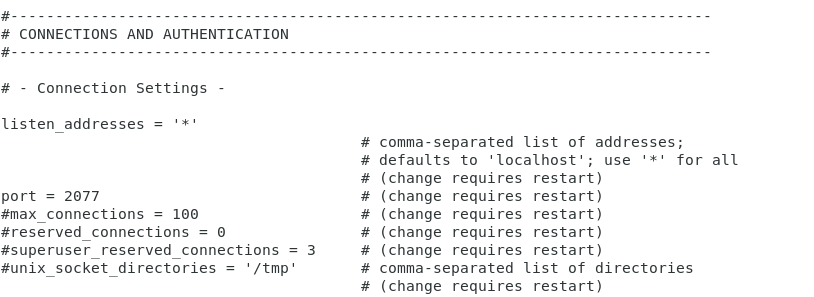


1. Modificación de archivo de configuración y validar que las modificaciones sean válidas.

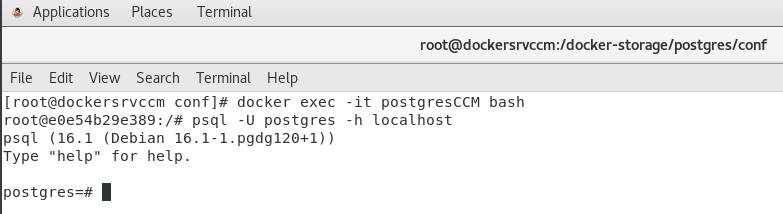


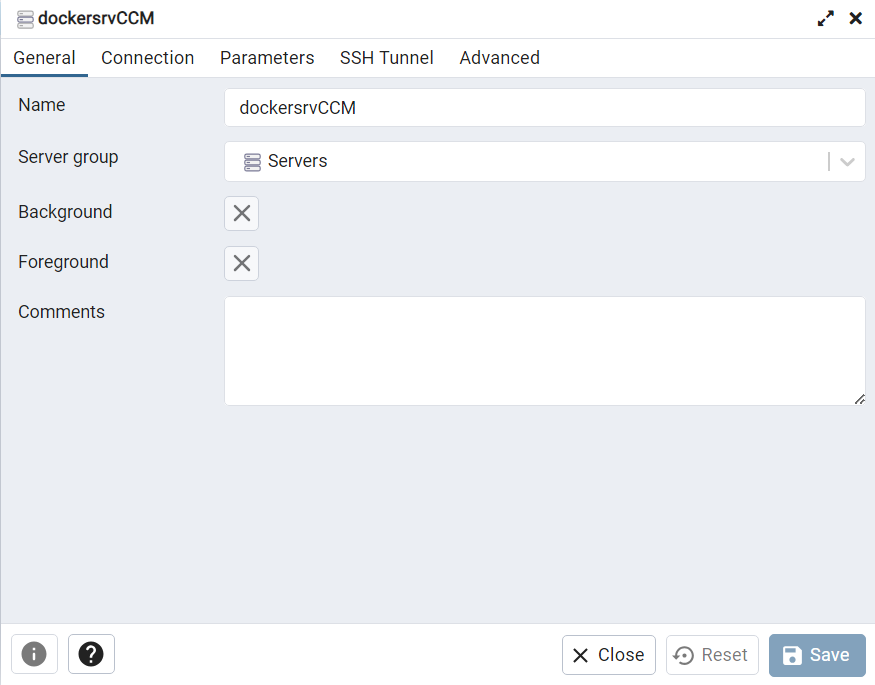


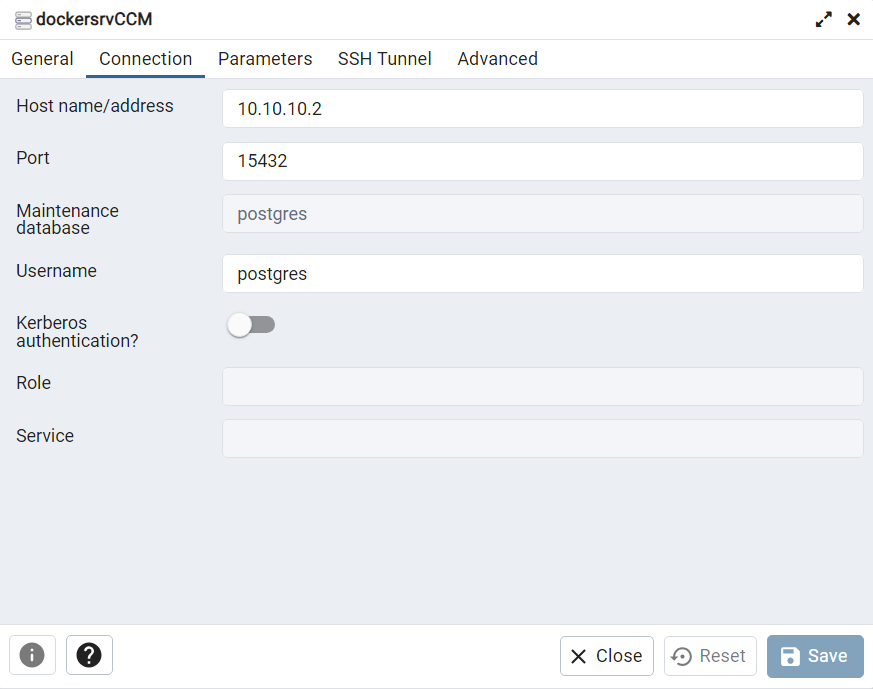
1. No inclusión del puerto de conexión por default y mapeo del puerto de conexión de la instancia de base de datos hacia el host

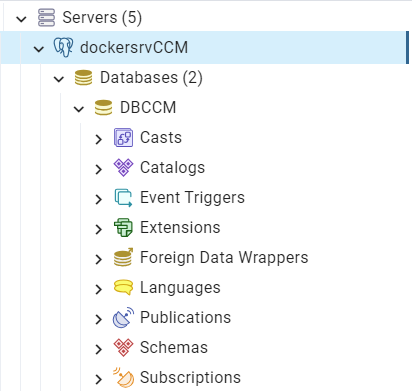


1. Uso de PGadmin para el acceso y manejo de instancia de base de datos contenerizada.
2. Validar conexión a Postgres a través de modo interactivo en bash y cliente gráfico





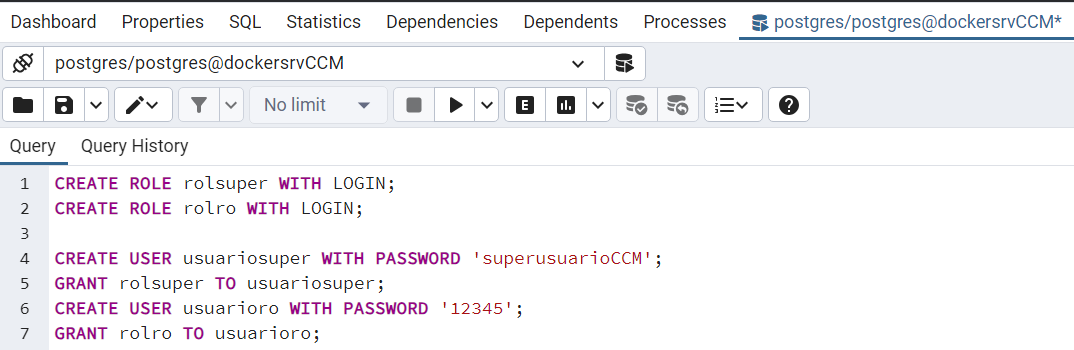


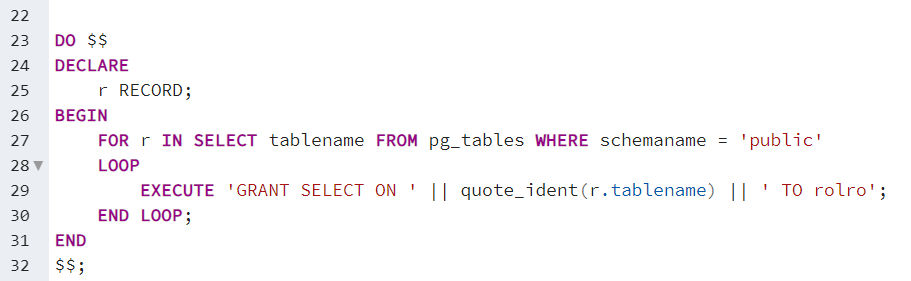


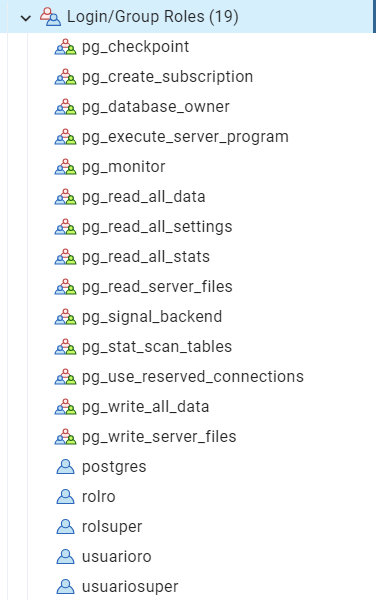
1. Creación de roles y usuarios:

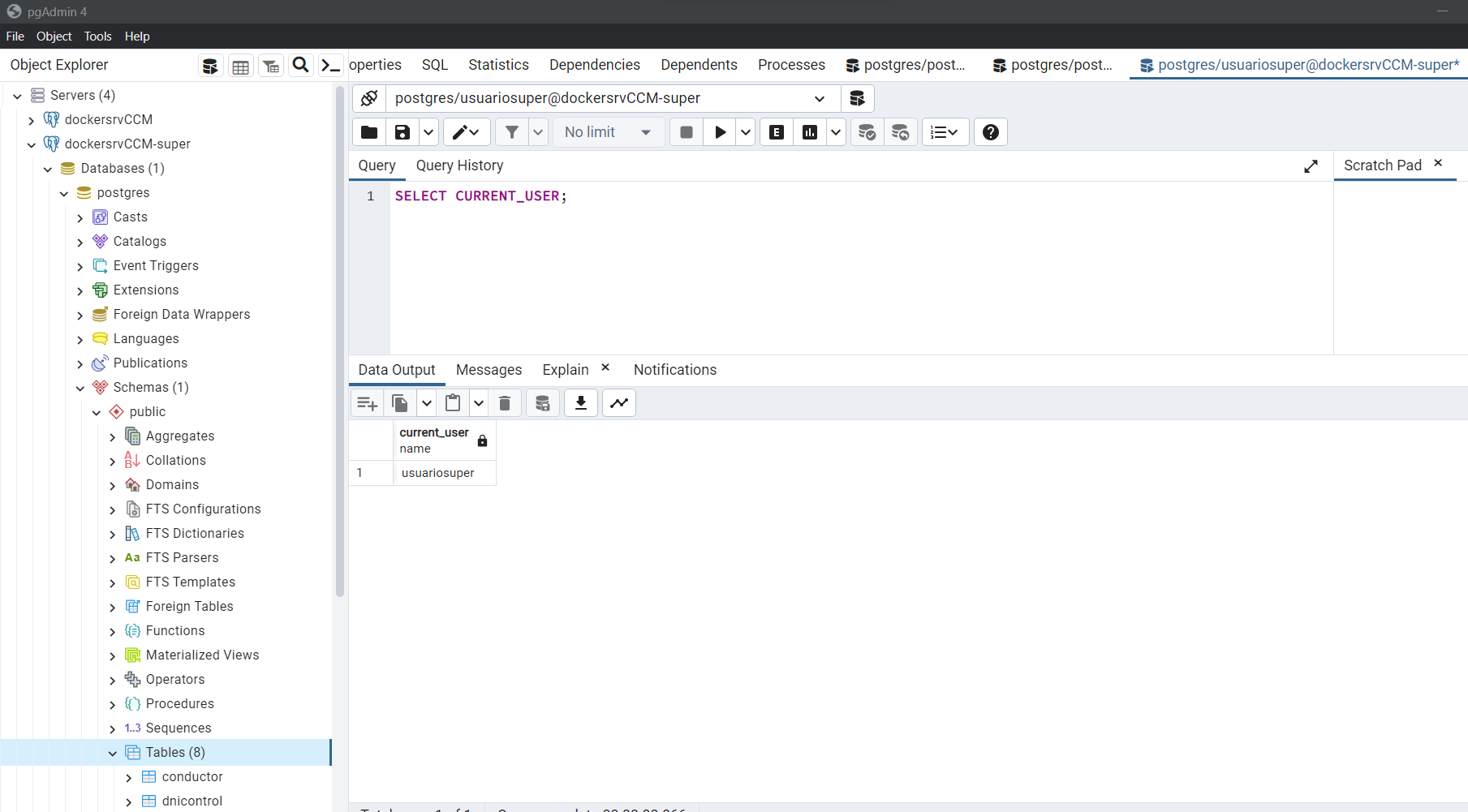
A continuación, se presentan usuarios creados, los roles asignados a ellos respectivamente y los privilegios de cada rol.

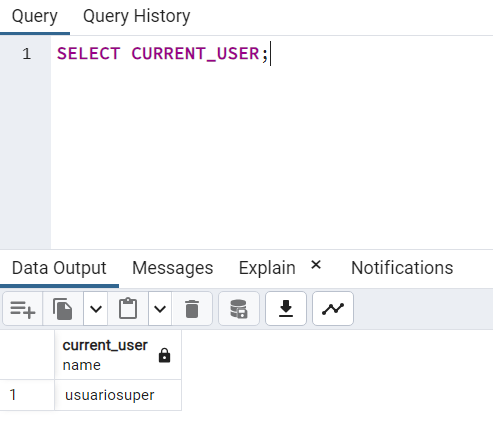
|  |  |  |
| --- | --- | --- |
| **USUARIOS** | **ROLES** | **PRIVILEGIOS** |
| **usuarioro** | **rolro** | **SELECT** |
| **usuariosuper** | **rolsuper** | **SELECT, INSERT, UPDATE, DELETE** |

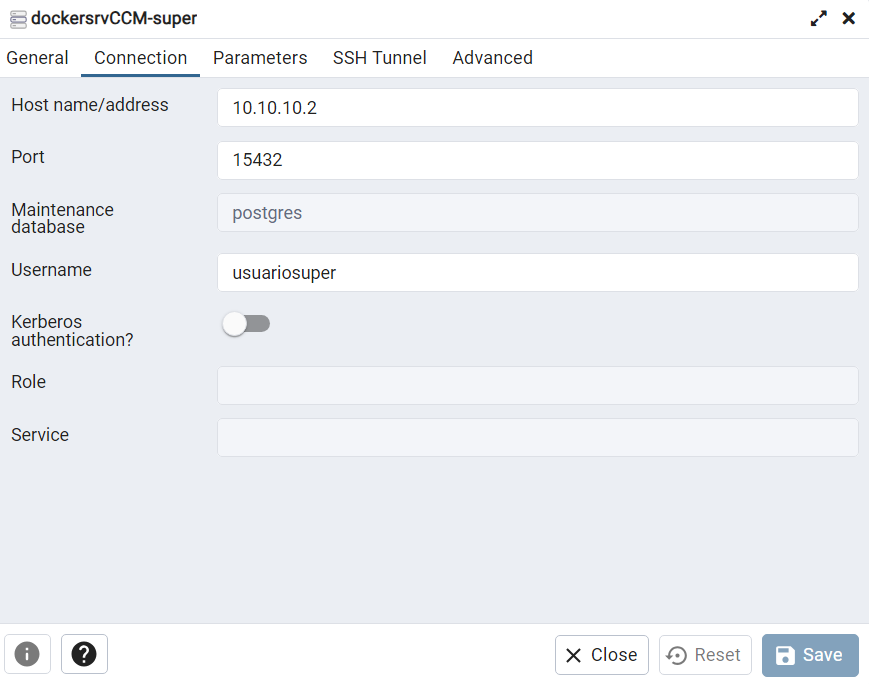


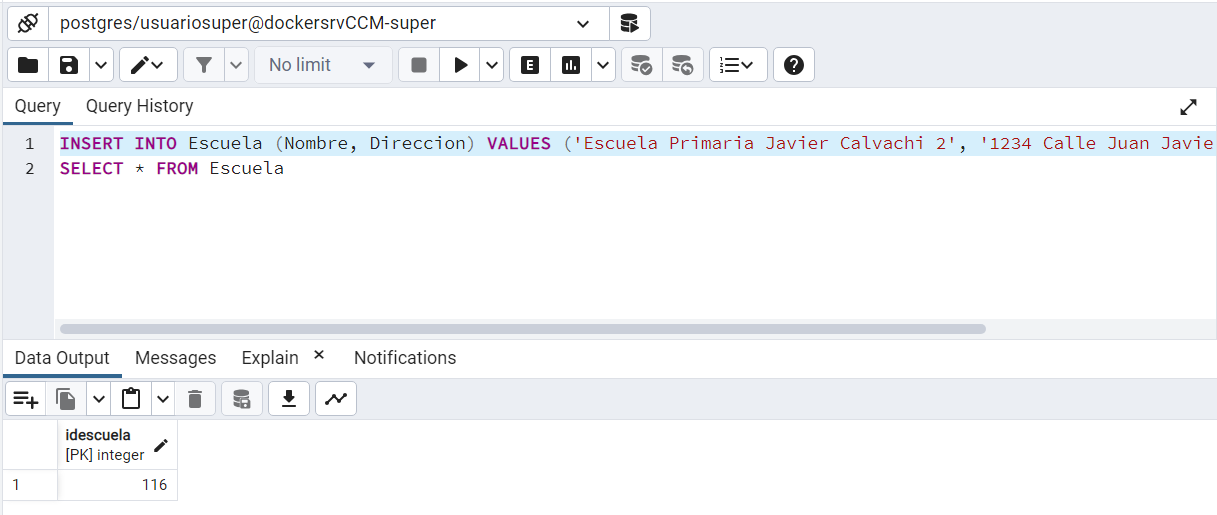


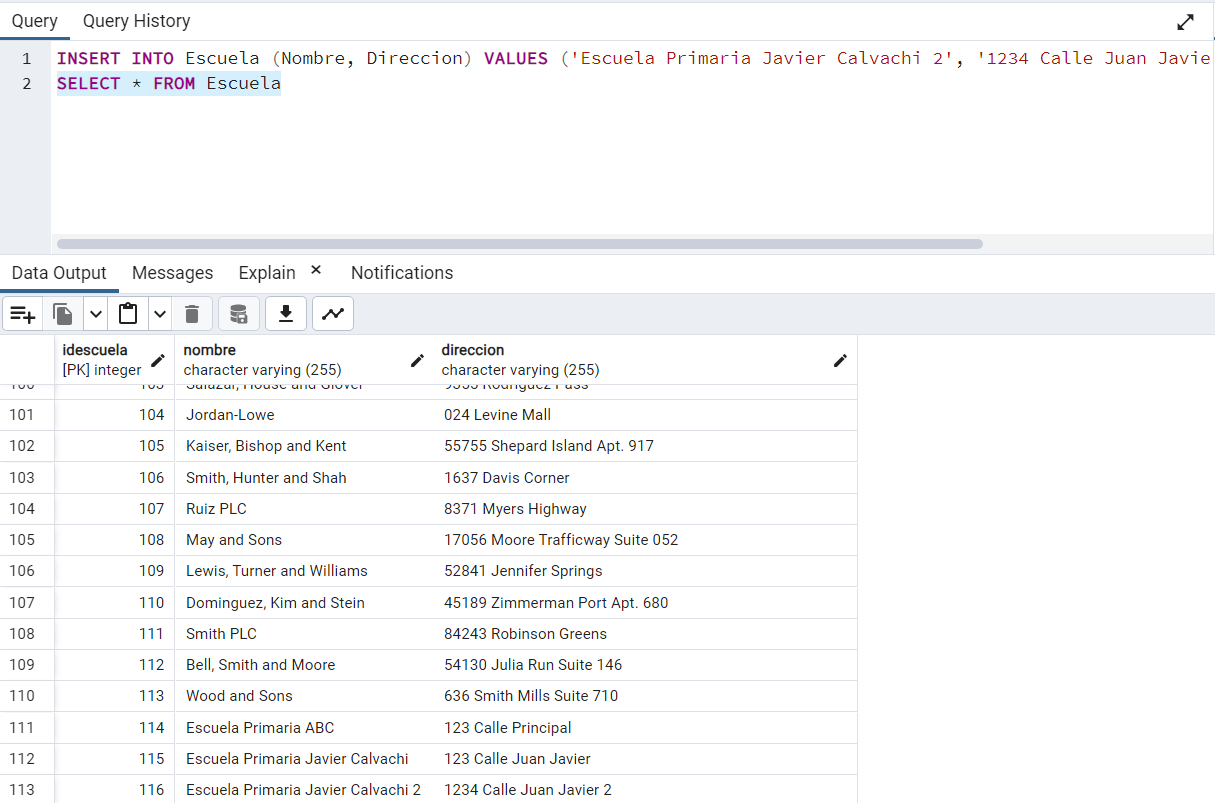


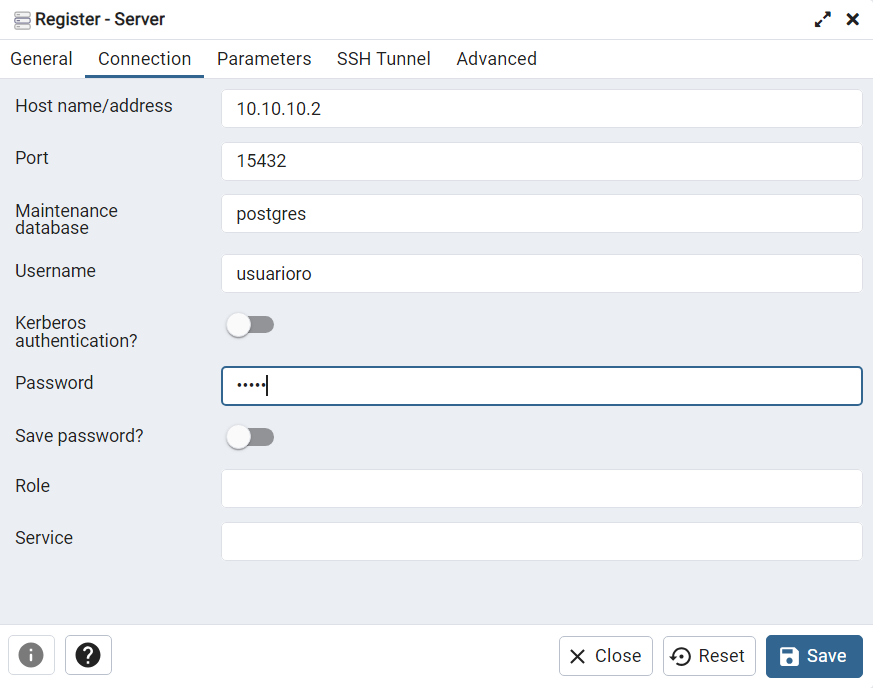


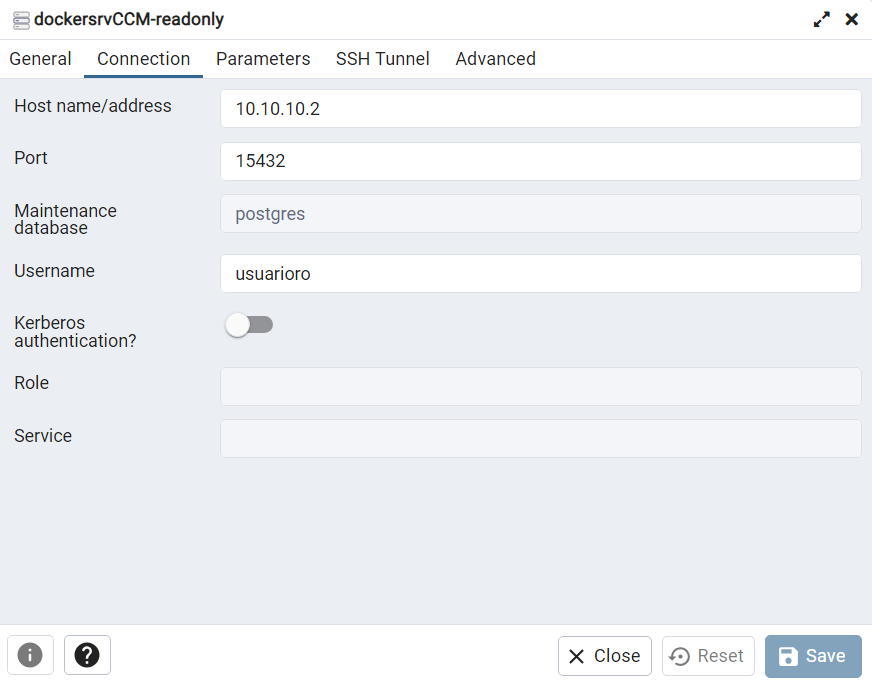


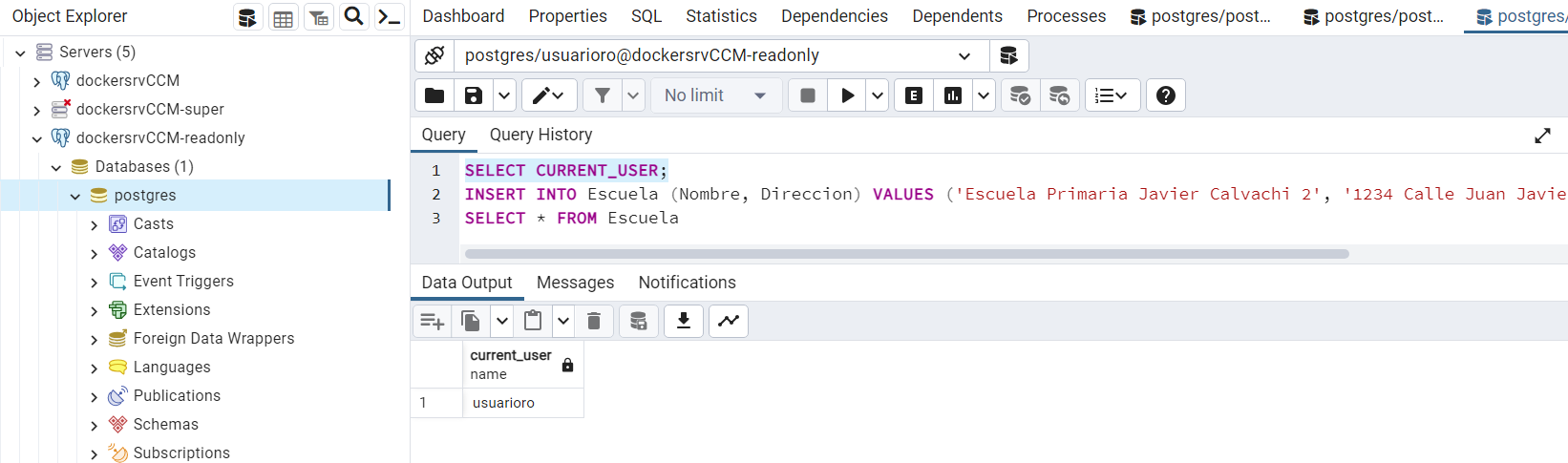


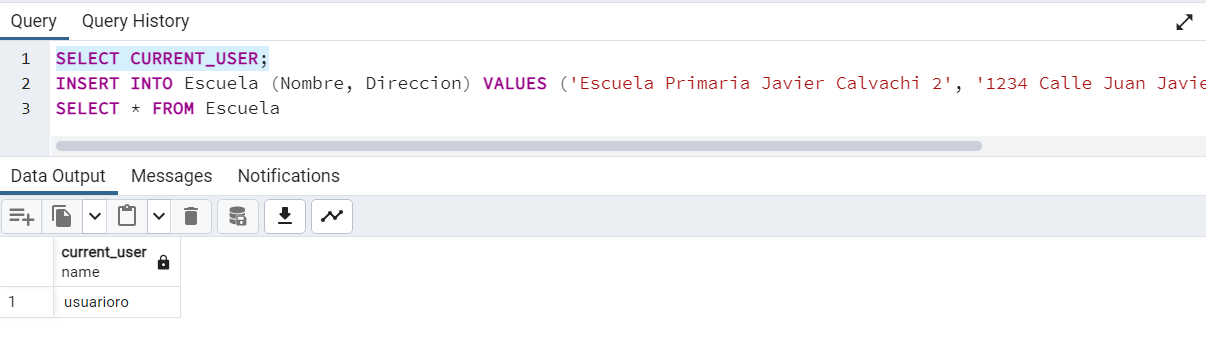


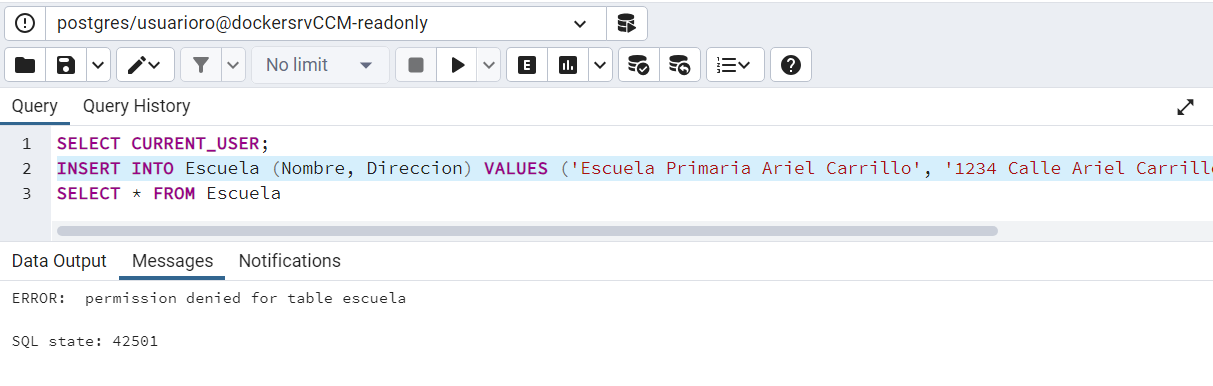


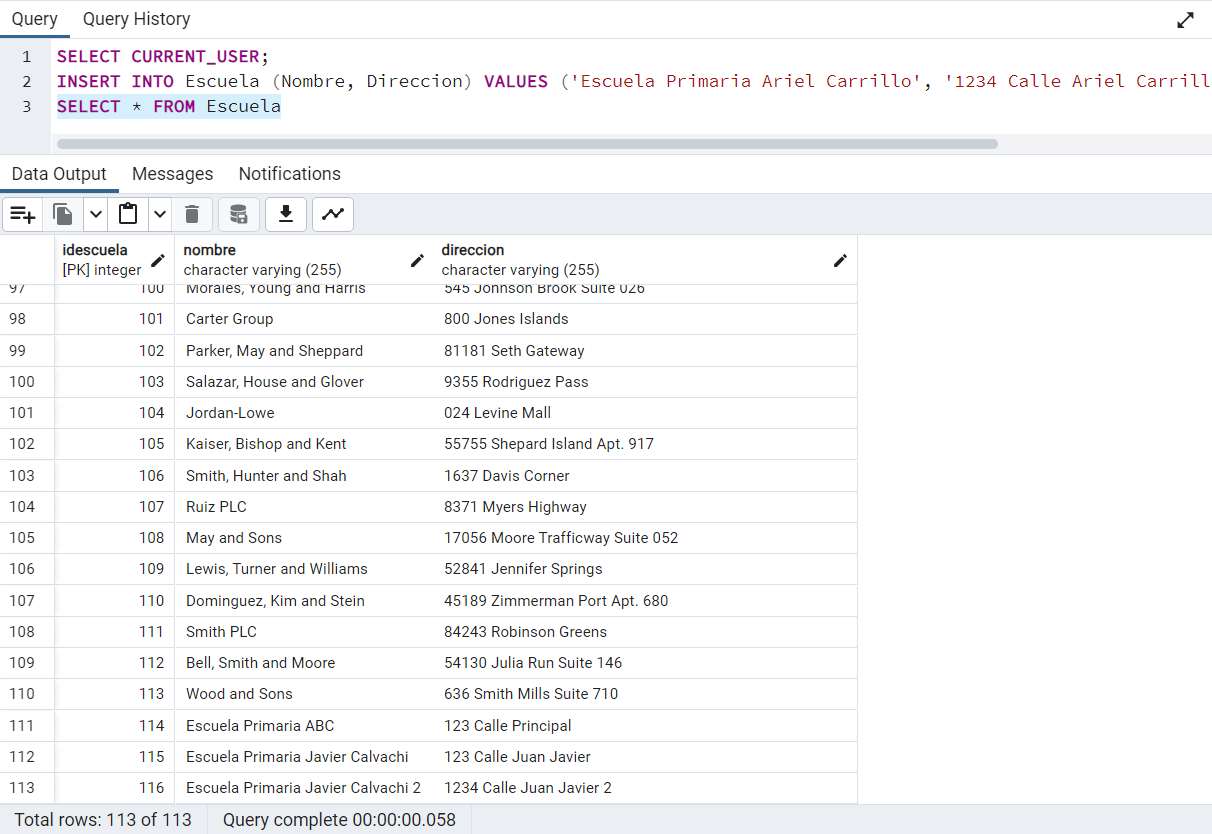


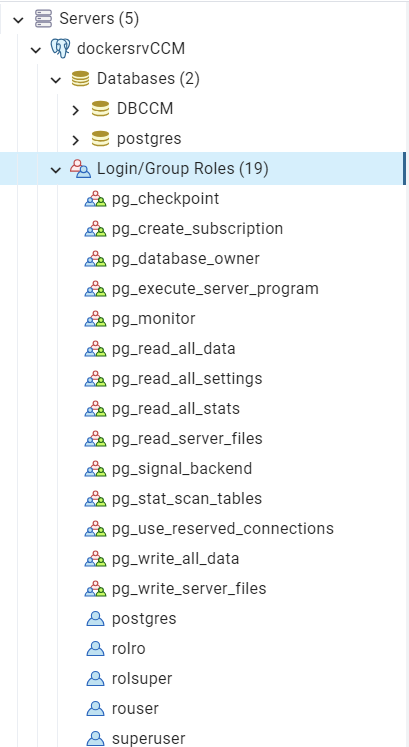


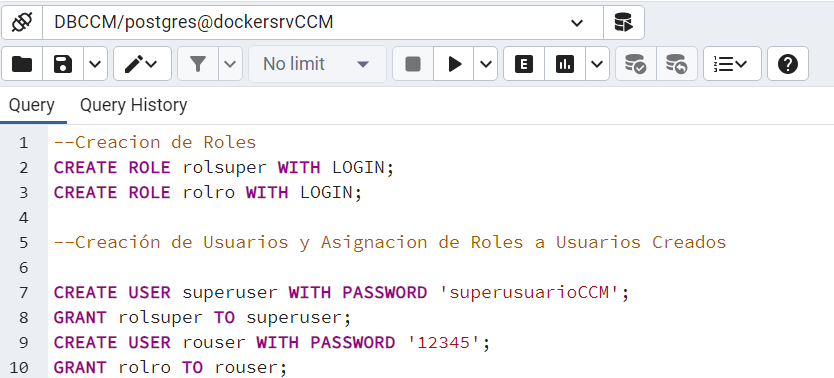


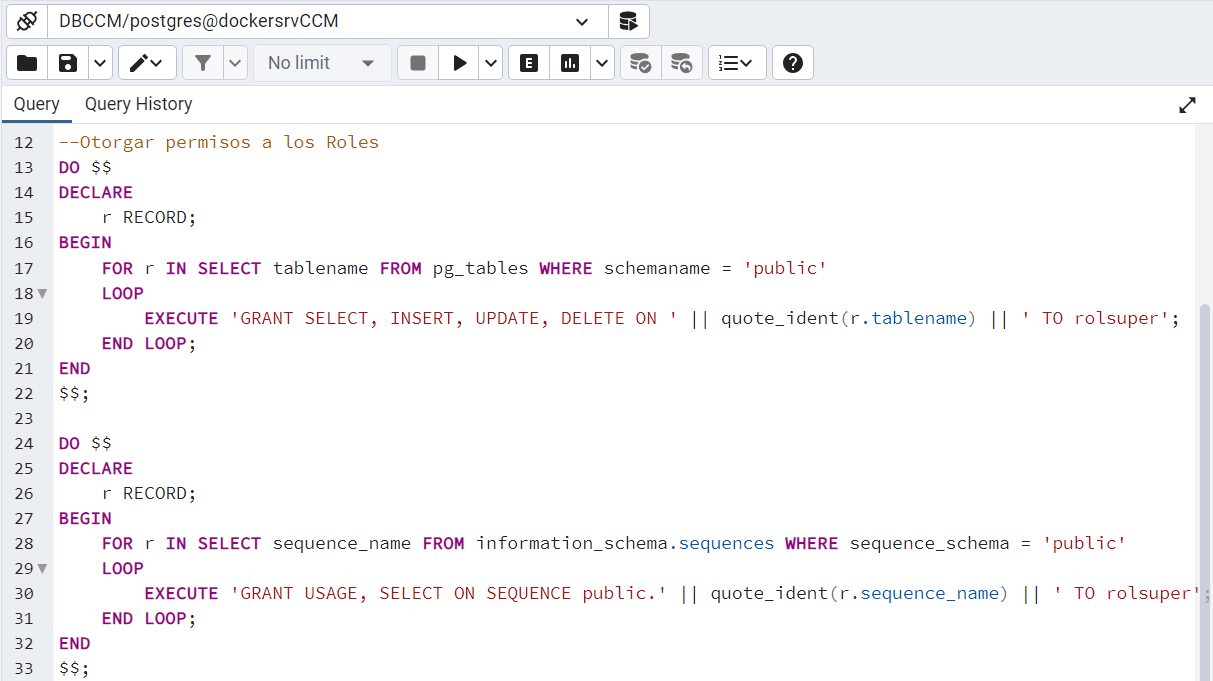


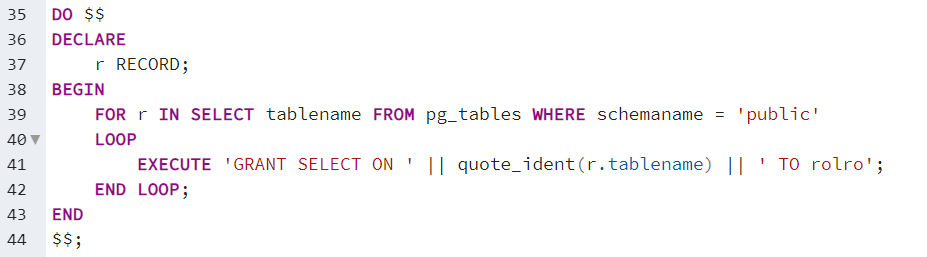


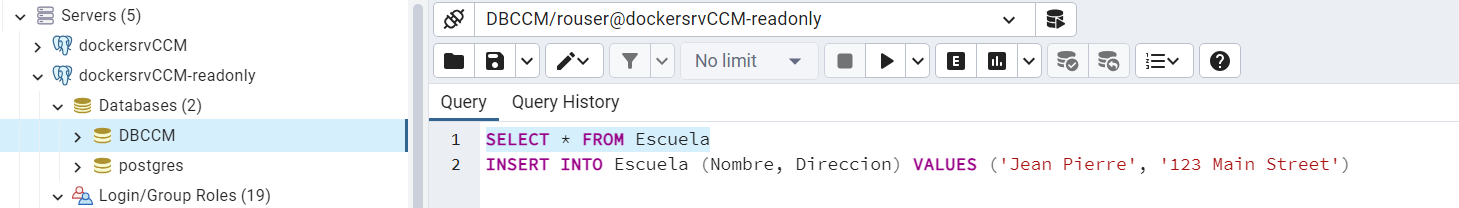


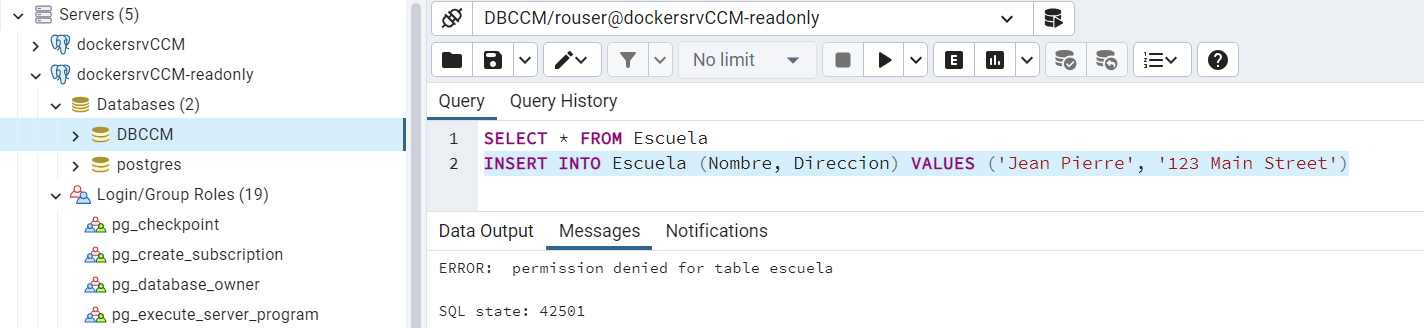


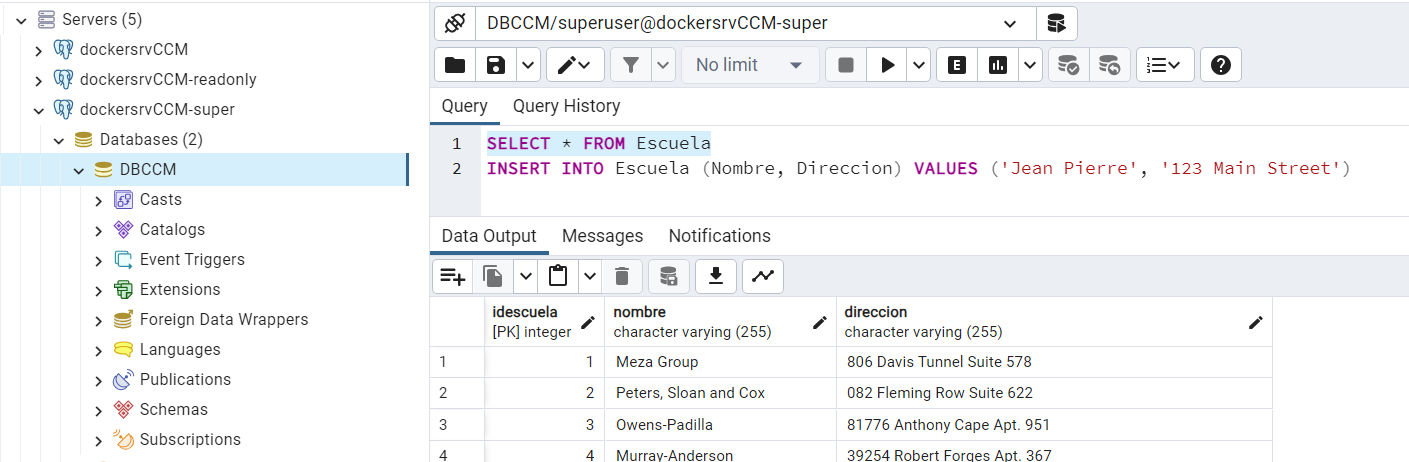


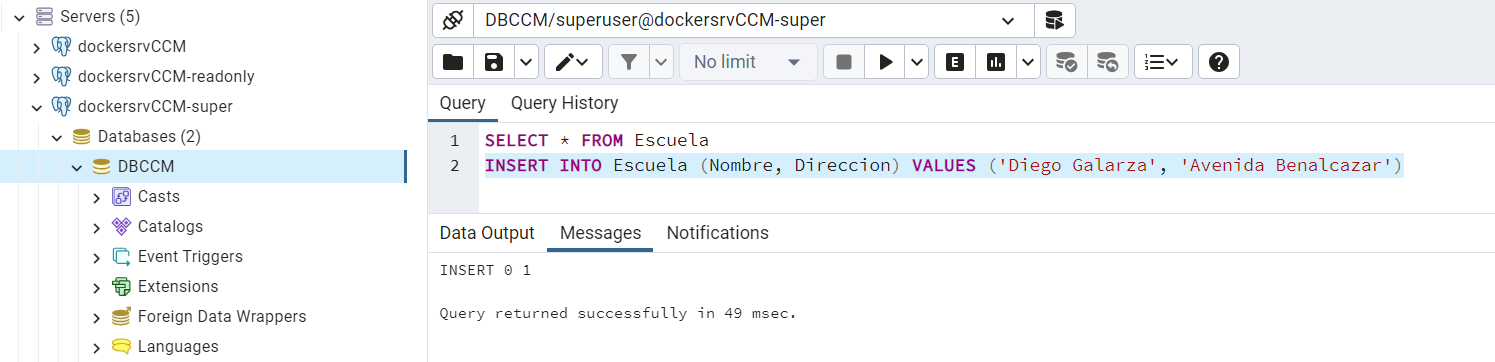


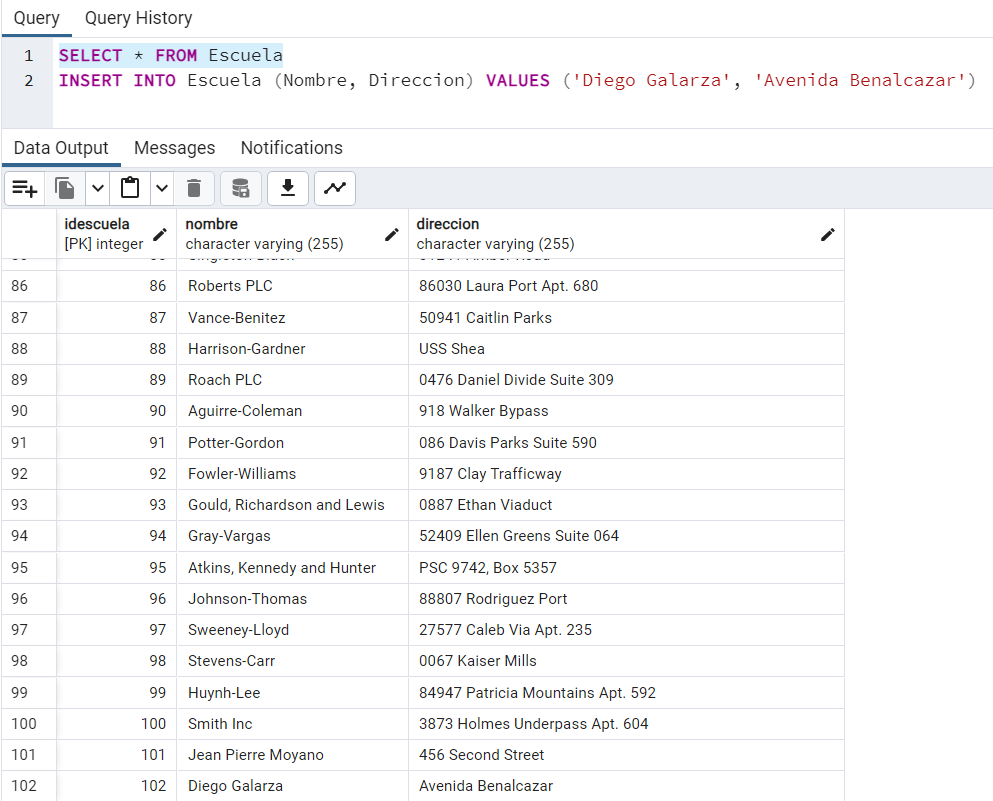


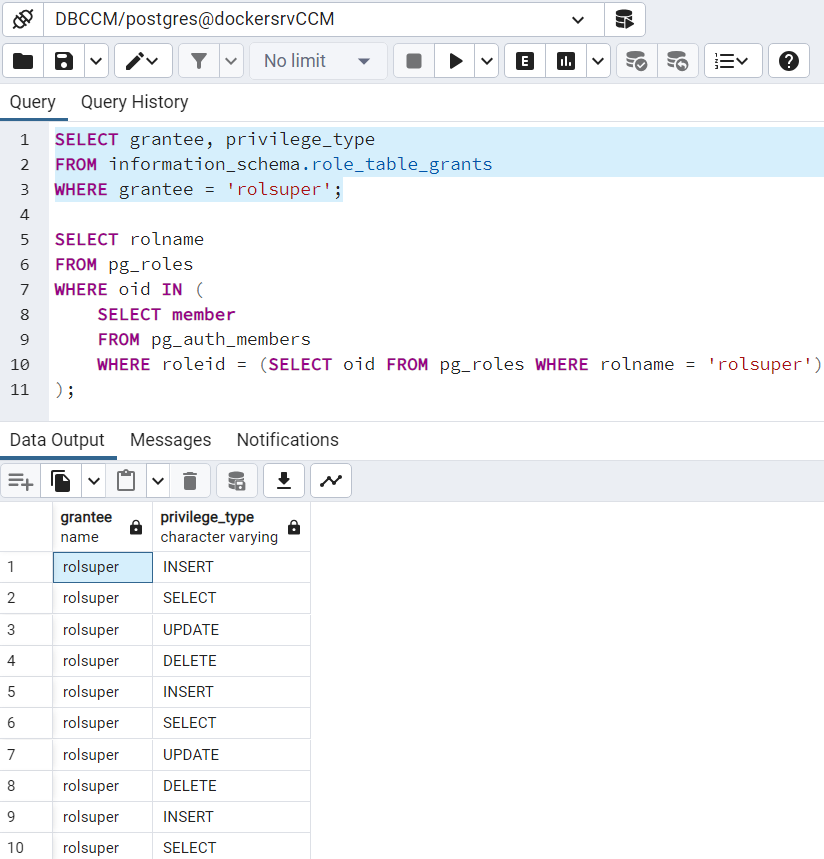


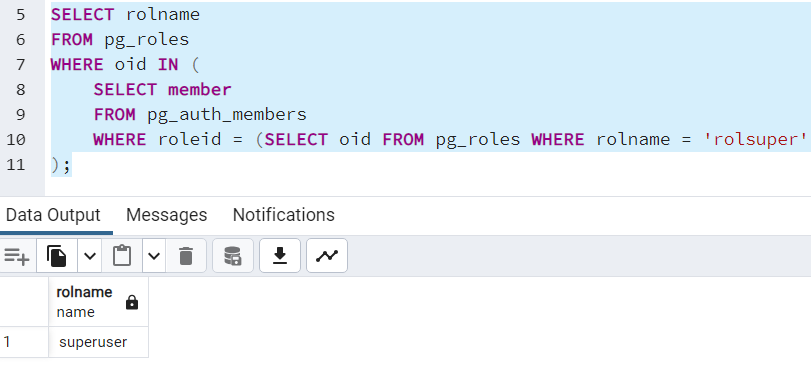


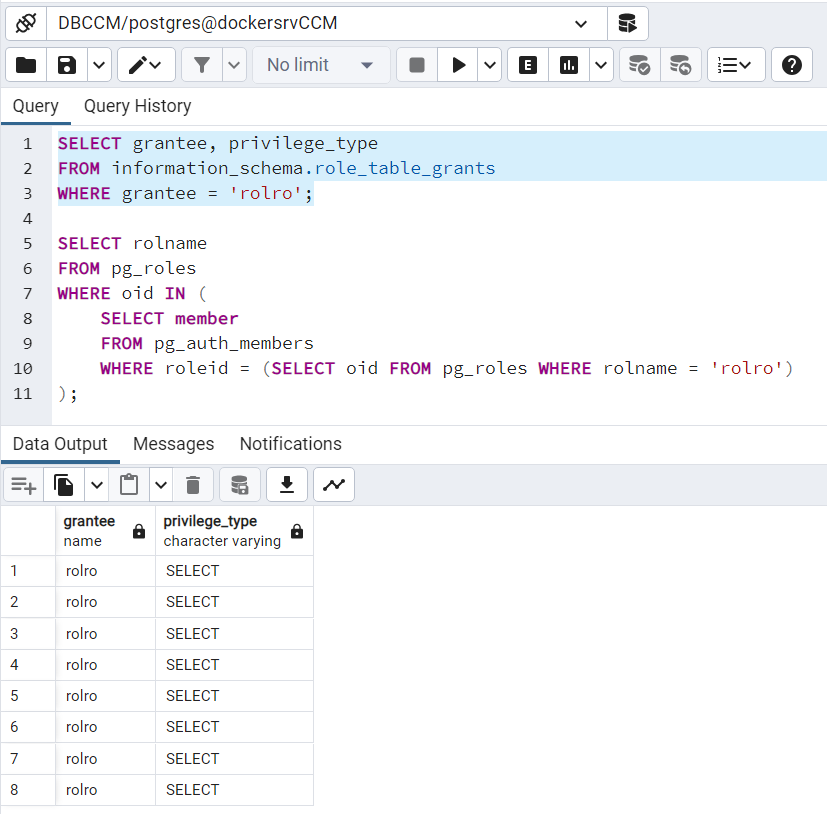


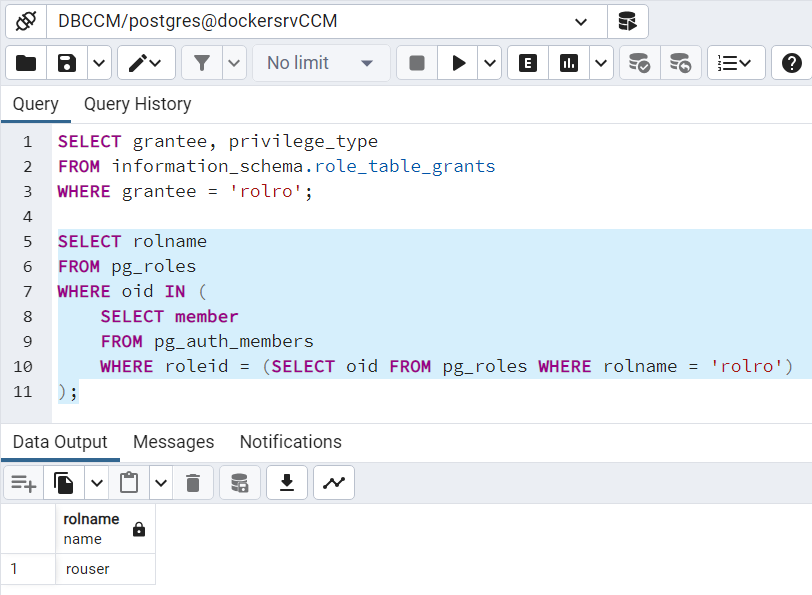






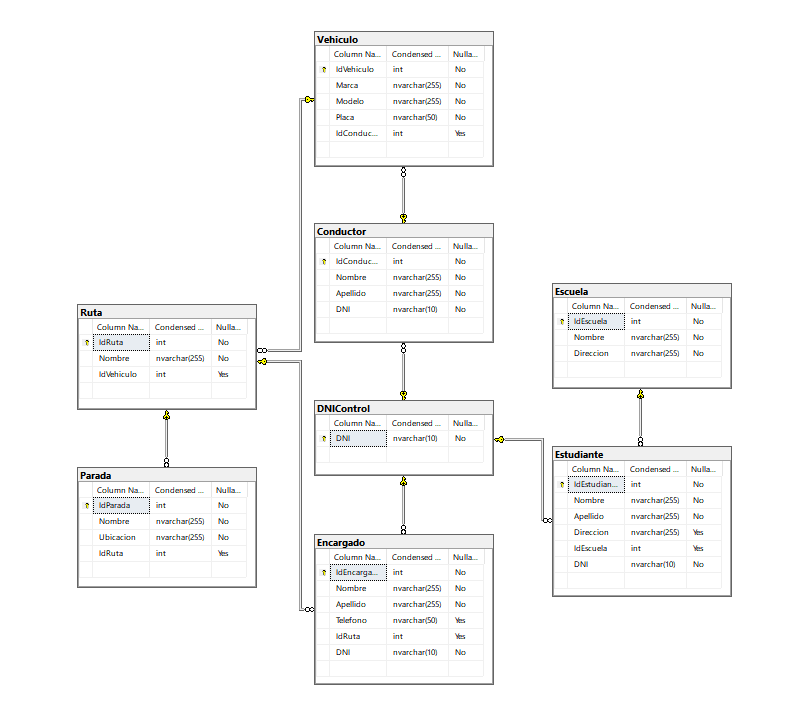






1. Diseño de la base de datos y diccionario de datos

El diseño con un diagrama físico de la base de datos y el diseño del diccionario de datos brinda una representación visual y una descripción detallada de la estructura y las relaciones de los datos

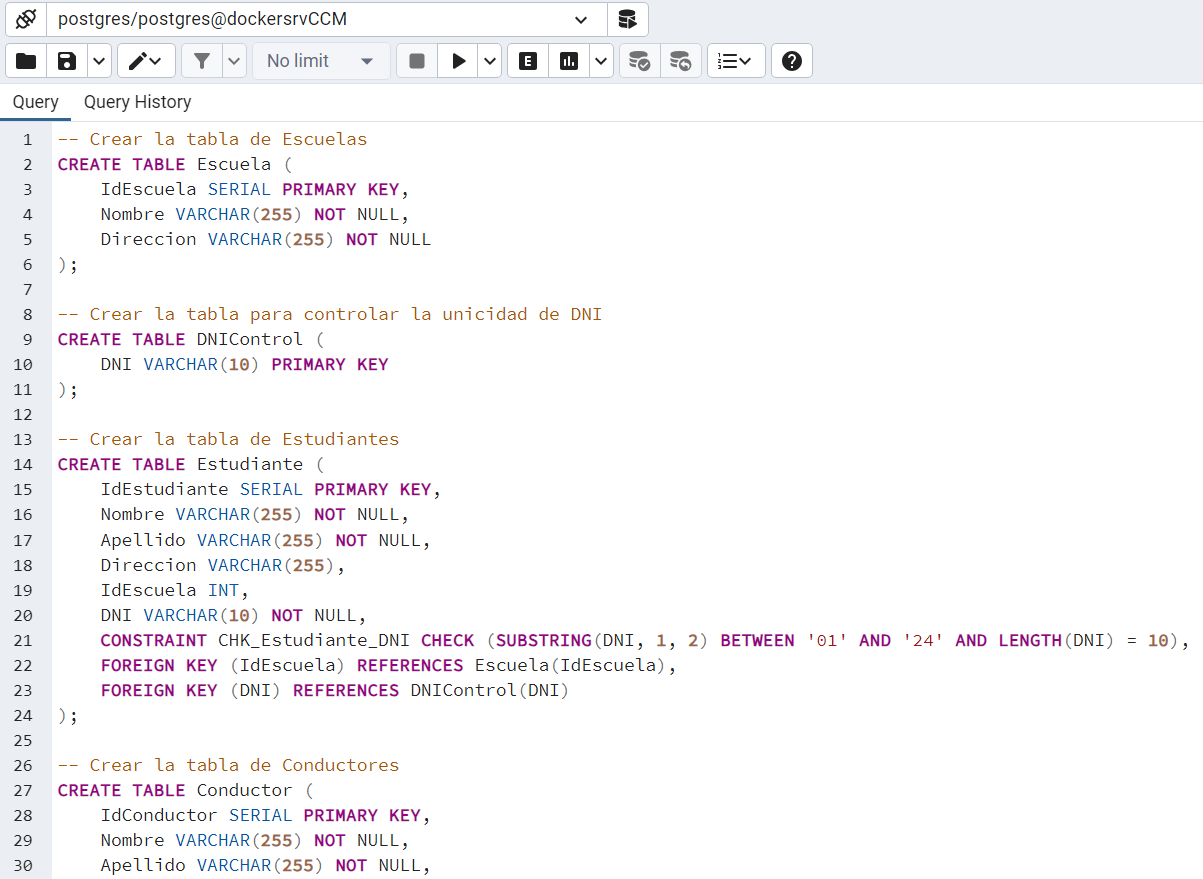


**Diccionario de Datos:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tabla** | **Campo** | **Tipo de Dato** | **Clave** | **Restricciones y Relaciones** | **Descripción** |
| Escuela | IdEscuela | SERIAL | PK | - | Identificador único autoincremental para cada escuela. |
| Nombre | VARCHAR(255) | - | NOT NULL | Nombre de la escuela. |
| Direccion | VARCHAR(255) | - | NOT NULL | Dirección de la escuela. |
| DNIControl | DNI | VARCHAR(10) | PK | - | DNI del individuo, usado para controlar la unicidad. |
| Estudiante | IdEstudiante | SERIAL | PK | - | Identificador único autoincremental para cada estudiante. |
| Nombre | VARCHAR(255) | - | NOT NULL | Nombre del estudiante. |
| Apellido | VARCHAR(255) | - | NOT NULL | Apellido del estudiante. |
| Direccion | VARCHAR(255) | - | - | Dirección del estudiante. |
| IdEscuela | INT | FK | FOREIGN KEY REFERENCES Escuela(IdEscuela) | Clave foránea que referencia a la tabla Escuela. |
| DNI | VARCHAR(10) | FK | NOT NULL, FOREIGN KEY REFERENCES DNIControl(DNI) | DNI del estudiante, controlado por la tabla DNIControl. |
| Conductor | IdConductor | SERIAL | PK | - | Identificador único autoincremental para cada conductor. |
| Nombre | VARCHAR(255) | - | NOT NULL | Nombre del conductor. |
| Apellido | VARCHAR(255) | - | NOT NULL | Apellido del conductor. |
| DNI | VARCHAR(10) | FK | NOT NULL, FOREIGN KEY REFERENCES DNIControl(DNI) | DNI del conductor, controlado por la tabla DNIControl. |
| Vehiculo | IdVehiculo | SERIAL | PK | - | Identificador único autoincremental para cada vehículo. |
| Marca | VARCHAR(255) | - | NOT NULL | Marca del vehículo. |
| Modelo | VARCHAR(255) | - | NOT NULL | Modelo del vehículo. |
| Placa | VARCHAR(50) | - | NOT NULL, UNIQUE | Placa del vehículo, debe ser única. |
| IdConductor | INT | FK | FOREIGN KEY REFERENCES Conductor(IdConductor) | Clave foránea que referencia a la tabla Conductor. |
| Ruta | IdRuta | SERIAL | PK | - | Identificador único autoincremental para cada ruta. |
| Nombre | VARCHAR(255) | - | NOT NULL | Nombre de la ruta. |
| IdVehiculo | INT | FK | FOREIGN KEY REFERENCES Vehiculo(IdVehiculo) | Clave foránea que referencia a la tabla Vehículo |
|  |  |  |  |  |
| Parada | IdParada | SERIAL | PK | - | Identificador único autoincremental para cada parada. |
| Nombre | VARCHAR(255) | - | NOT NULL | Nombre de la parada. |
| Ubicacion | VARCHAR(255) | - | NOT NULL | Ubicación de la parada. |
| IdRuta | INT | FK | FOREIGN KEY REFERENCES Ruta(IdRuta) | Clave foránea que referencia a la tabla Ruta. |
| Encargado | IdEncargado | SERIAL | PK | - | Identificador único autoincremental para cada encargado. |
| Nombre | VARCHAR(255) | - | NOT NULL | Nombre del encargado. |
| Apellido | VARCHAR(255) | - | NOT NULL | Apellido del encargado. |
| Telefono | VARCHAR(50) | - | - | Teléfono del encargado. |
| IdRuta | INT | FK | FOREIGN KEY REFERENCES Ruta(IdRuta) | Clave foránea que referencia a la tabla Ruta. |
| DNI | VARCHAR(10) | FK | NOT NULL, FOREIGN KEY REFERENCES DNIControl(DNI) | DNI del encargado, controlado por la tabla DNIControl. |

**Creación de Tablas en la Base de Datos:**

1. Creación de Tablas



Script utilizado:

-- Crear la tabla de Escuelas

CREATE TABLE Escuela (

IdEscuela SERIAL PRIMARY KEY,

Nombre VARCHAR(255) NOT NULL,

Direccion VARCHAR(255) NOT NULL

);

-- Crear la tabla para controlar la unicidad de DNI

CREATE TABLE DNIControl (

DNI VARCHAR(10) PRIMARY KEY

);

-- Crear la tabla de Estudiantes

CREATE TABLE Estudiante (

IdEstudiante SERIAL PRIMARY KEY,

Nombre VARCHAR(255) NOT NULL,

Apellido VARCHAR(255) NOT NULL,

Direccion VARCHAR(255),

IdEscuela INT,

DNI VARCHAR(10) NOT NULL,

CONSTRAINT CHK\_Estudiante\_DNI CHECK (SUBSTRING(DNI, 1, 2) BETWEEN '01' AND '24' AND LENGTH(DNI) = 10),

FOREIGN KEY (IdEscuela) REFERENCES Escuela(IdEscuela),

FOREIGN KEY (DNI) REFERENCES DNIControl(DNI)

);

-- Crear la tabla de Conductores

CREATE TABLE Conductor (

IdConductor SERIAL PRIMARY KEY,

Nombre VARCHAR(255) NOT NULL,

Apellido VARCHAR(255) NOT NULL,

DNI VARCHAR(10) NOT NULL,

CONSTRAINT CHK\_Conductor\_DNI CHECK (SUBSTRING(DNI, 1, 2) BETWEEN '01' AND '24' AND LENGTH(DNI) = 10),

FOREIGN KEY (DNI) REFERENCES DNIControl(DNI)

);

-- Crear la tabla de Vehículos

CREATE TABLE Vehiculo (

IdVehiculo SERIAL PRIMARY KEY,

Marca VARCHAR(255) NOT NULL,

Modelo VARCHAR(255) NOT NULL,

Placa VARCHAR(50) NOT NULL UNIQUE,

IdConductor INT,

FOREIGN KEY (IdConductor) REFERENCES Conductor(IdConductor)

);

-- Crear la tabla de Rutas

CREATE TABLE Ruta (

IdRuta SERIAL PRIMARY KEY,

Nombre VARCHAR(255) NOT NULL,

IdVehiculo INT,

FOREIGN KEY (IdVehiculo) REFERENCES Vehiculo(IdVehiculo)

);

-- Crear la tabla de Paradas

CREATE TABLE Parada (

IdParada SERIAL PRIMARY KEY,

Nombre VARCHAR(255) NOT NULL,

Ubicacion VARCHAR(255) NOT NULL,

IdRuta INT,

FOREIGN KEY (IdRuta) REFERENCES Ruta(IdRuta)

);

-- Crear la tabla de Encargados

CREATE TABLE Encargado (

IdEncargado SERIAL PRIMARY KEY,

Nombre VARCHAR(255) NOT NULL,

Apellido VARCHAR(255) NOT NULL,

Telefono VARCHAR(50),

IdRuta INT,

DNI VARCHAR(10) NOT NULL,

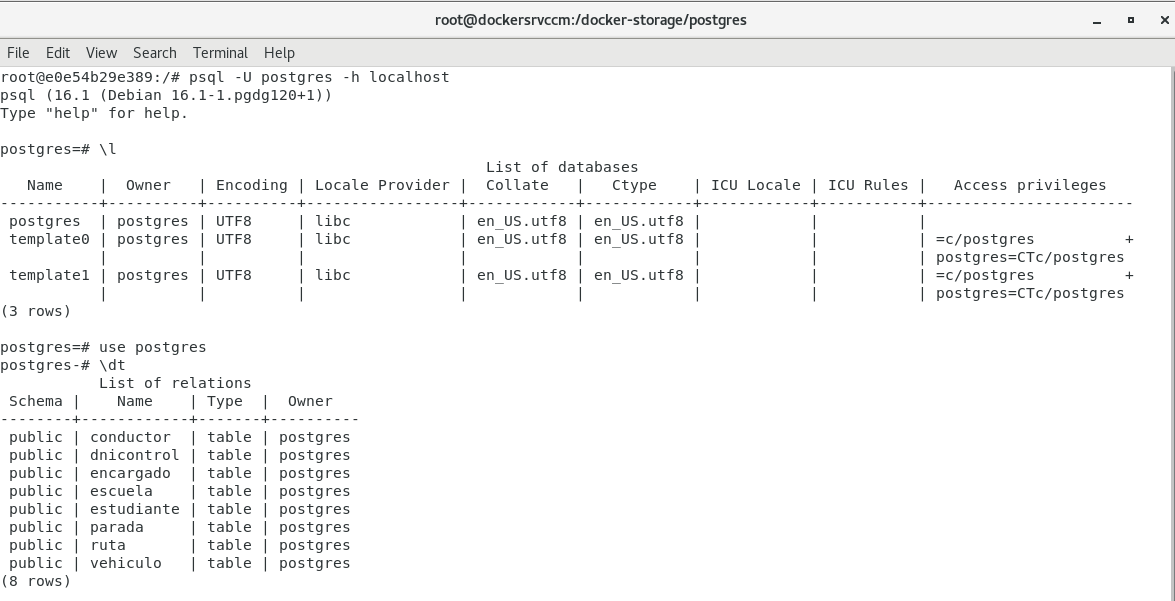
CONSTRAINT CHK\_Encargado\_DNI CHECK (SUBSTRING(DNI, 1, 2) BETWEEN '01' AND '24' AND LENGTH(DNI) = 10),

FOREIGN KEY (IdRuta) REFERENCES Ruta(IdRuta),

FOREIGN KEY (DNI) REFERENCES DNIControl(DNI)

);

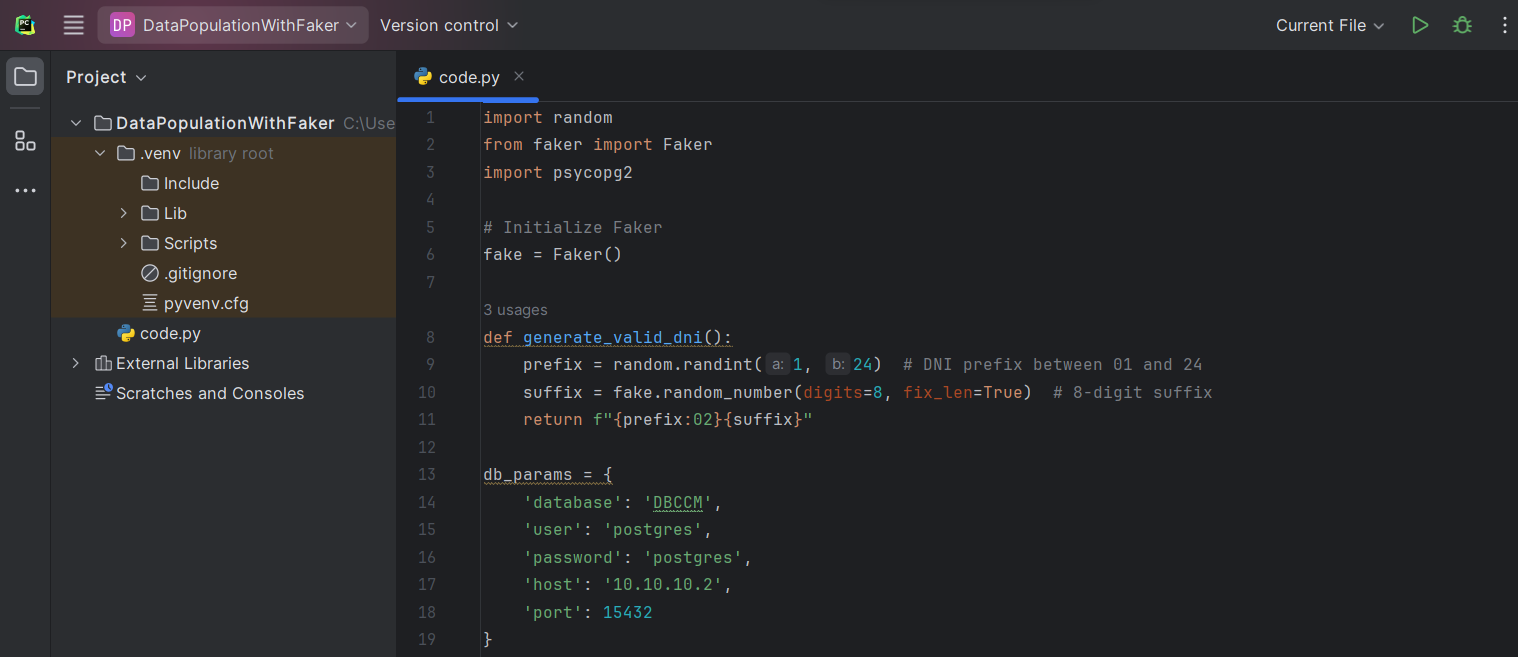




1. Generación del contenido de la base de datos usando Python y módulo Faker:

Código en Python:

import random  
from faker import Faker  
import psycopg2  
  
# Initialize Faker  
fake = Faker()  
  
def generate\_valid\_dni():  
 prefix = random.randint(1, 24) # DNI prefix between 01 and 24  
 suffix = fake.random\_number(digits=8, fix\_len=True) # 8-digit suffix  
 return f"{prefix:02}{suffix}"  
  
db\_params = {  
 'database': 'DBCCM',  
 'user': 'postgres',  
 'password': 'postgres',  
 'host': '10.10.10.2',  
 'port': 15432  
}  
  
def create\_fake\_data():  
  
 conn = psycopg2.connect(\*\*db\_params)  
 cursor = conn.cursor()  
  
 for \_ in range(100): # Adjust the range for the number of records  
 # Escuela  
 school\_name = fake.company()  
 school\_address = fake.address()  
 cursor.execute("INSERT INTO Escuela (Nombre, Direccion) VALUES (%s, %s) RETURNING IdEscuela", (school\_name, school\_address))  
 school\_id = cursor.fetchone()[0]  
  
 # DNIControl and Estudiante  
 student\_dni = generate\_valid\_dni()  
 cursor.execute("INSERT INTO DNIControl (DNI) VALUES (%s)", (student\_dni,))  
 student\_name = fake.first\_name()  
 student\_surname = fake.last\_name()  
 student\_address = fake.address()  
 cursor.execute("INSERT INTO Estudiante (Nombre, Apellido, Direccion, IdEscuela, DNI) VALUES (%s, %s, %s, %s, %s)",  
 (student\_name, student\_surname, student\_address, school\_id, student\_dni))  
  
 # Conductor  
 driver\_dni = generate\_valid\_dni()  
 cursor.execute("INSERT INTO DNIControl (DNI) VALUES (%s)", (driver\_dni,))  
 driver\_name = fake.first\_name()  
 driver\_surname = fake.last\_name()  
 cursor.execute("INSERT INTO Conductor (Nombre, Apellido, DNI) VALUES (%s, %s, %s) RETURNING IdConductor",  
 (driver\_name, driver\_surname, driver\_dni))  
 driver\_id = cursor.fetchone()[0]  
  
 # Vehiculo  
 vehicle\_brand = fake.company()  
 vehicle\_model = fake.word()  
 vehicle\_plate = fake.bothify(text='??-###-##', letters='ABCDEFGHIJKLMNOPQRSTUVWXYZ')  
 cursor.execute("INSERT INTO Vehiculo (Marca, Modelo, Placa, IdConductor) VALUES (%s, %s, %s, %s) RETURNING IdVehiculo",  
 (vehicle\_brand, vehicle\_model, vehicle\_plate, driver\_id))  
 vehicle\_id = cursor.fetchone()[0]  
  
 # Ruta  
 route\_name = fake.street\_name()  
 cursor.execute("INSERT INTO Ruta (Nombre, IdVehiculo) VALUES (%s, %s) RETURNING IdRuta",  
 (route\_name, vehicle\_id))  
 route\_id = cursor.fetchone()[0]  
  
 # Parada  
 stop\_name = fake.street\_address()  
 stop\_location = fake.address()  
 cursor.execute("INSERT INTO Parada (Nombre, Ubicacion, IdRuta) VALUES (%s, %s, %s)",  
 (stop\_name, stop\_location, route\_id))  
  
 # Encargado  
 manager\_dni = generate\_valid\_dni()  
 cursor.execute("INSERT INTO DNIControl (DNI) VALUES (%s)", (manager\_dni,))  
 manager\_name = fake.first\_name()  
 manager\_surname = fake.last\_name()  
 manager\_phone = fake.phone\_number()  
 cursor.execute("INSERT INTO Encargado (Nombre, Apellido, Telefono, IdRuta, DNI) VALUES (%s, %s, %s, %s, %s)",  
 (manager\_name, manager\_surname, manager\_phone, route\_id, manager\_dni))  
  
 conn.commit()  
 cursor.close()  
 conn.close()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 create\_fake\_data()



1. **REPOSITORIO GITHUB**
2. **CUMPLIMIENTO DE CRITERIOS DE EVALUACION**

|  |  |  |
| --- | --- | --- |
| Criterio | Porcentaje de Cumplimiento | Evidencia(s) |
| Definición de los Objetivos de Trabajo | 100% | * **Mejorar el rendimiento de la base de datos para que los procesos sean más eficientes al dar respuesta.** * **Analizar la confidencialidad, integridad y disponibilidad de los datos almacenados, considerando RBAC.** * **Adaptar el crecimiento empresarial en función de la cantidad de datos a gestionar mediante políticas de seguridad.** |
| Definición de Cronograma | 100% |  |
| Definición de Roles | 100% |  |
| Asignación de Roles | 100% |  |
| Asignación de Responsabilidades | 100% |  |
| Cronograma de reuniones de trabajo | 190% | 1. Primera reunión (10 de diciembre 2023 a las 14h00 pm)      1. Segunda reunión (7 de enero del 2024 a las 15h47)      1. Tercera reunión (12 de enero del 2024 a las 22h04) |
| Ideas aportadas para la implementación de la solución por cada participante | 100% |  |
| Aporte individual para la consecución de los Objetivos de Trabajo | 100% |  |