

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS, 2024-II
FUNDAMENTOS DE BASES DE DATOS

BITÁCORA, PRÁCTICA 01 INSTALACIÓN DE DOCKER Y DEL SMBD, POSTGRESQL

EQUIPO: CHIQUESSQL

DANIEL ROJO MATA
314297967

PROFESOR:

GERARDO ÁVILES ROSAS

AYUDANTES DE TEORÍA:

GERARDO URIEL SOTO MIRANDA
VALERIA FERNANDA MANJARREZ ANGELES

AYUDANTES DE LABORATORIO:

RICARDO BADILLO MACÍAS
ROCÍO AYLIN HUERTA GONZÁLEZ

I. DETALLES DEL SISTEMA

A continuación, se presentan los pasos seguidos durante la instalación de Docker. La versión del software instalado es Docker 25.0.3, build 4deb41.

Es esencial destacar que el sistema operativo utilizado fue Ubuntu en su versión 22.04.3, clasificada como una versión de soporte a largo plazo (LTS).

Se tomaron capturas de pantalla en cada etapa del proceso de instalación. A pesar de ello, se realizaron recortes de manera que se destacara el comando y algunas líneas ejecutadas en la terminal en cada captura.

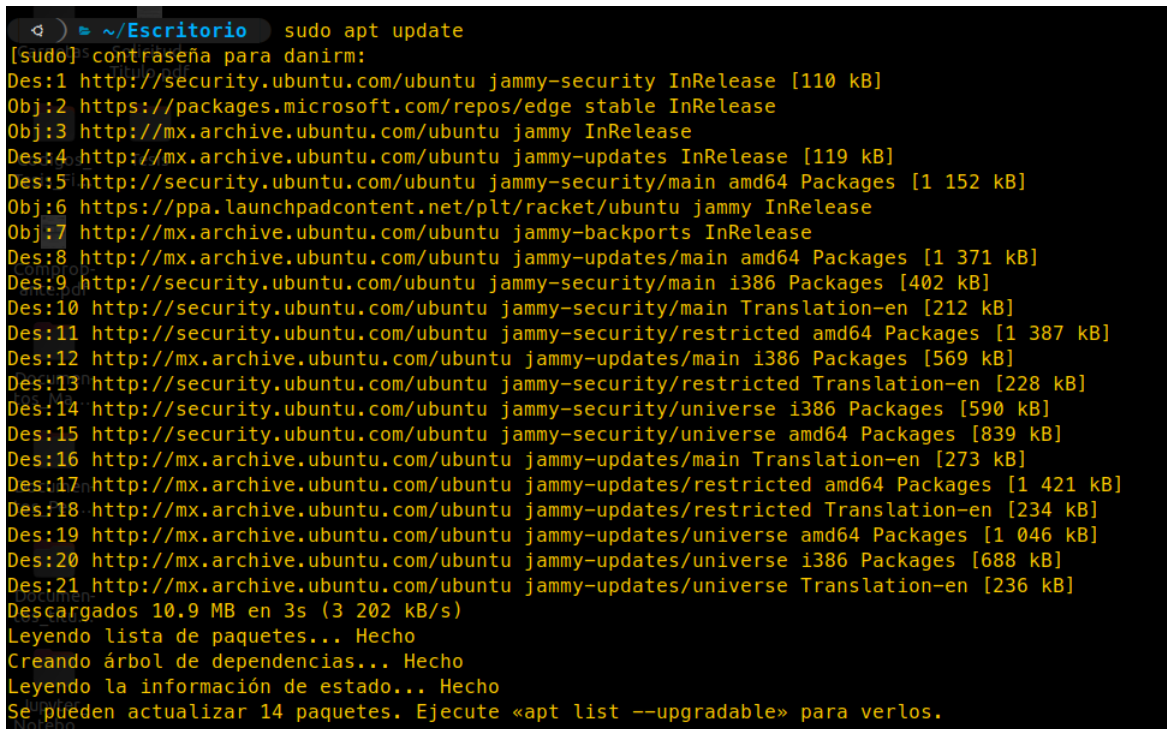
II. PROCESO DE INSTALACIÓN DE DOCKER

1. El primer paso es actualizar la lista de paquetes existentes, por lo que se ejecuta el siguiente comando en la terminal del sistema:

```
sudo apt update
```

Lo anterior se hace debido a que antes de instalar cualquier paquete, es importante asegurarse de tener la información de paquetes más reciente para evitar problemas de dependencias y obtener las últimas versiones de software.

Al ejecutarse en la terminal del sistema se tiene algo del siguiente estilo:



```
< ~/Escritorio sudo apt update
[sudo] contraseña para danirm:
Des:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Obj:2 https://packages.microsoft.com/repos/edge stable InRelease
Obj:3 http://mx.archive.ubuntu.com/ubuntu jammy InRelease
Des:4 http://mx.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Des:5 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1 152 kB]
Obj:6 https://ppa.launchpadcontent.net/plt/racket/ubuntu jammy InRelease
Obj:7 http://mx.archive.ubuntu.com/ubuntu jammy-backports InRelease
Des:8 http://mx.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1 371 kB]
Des:9 http://security.ubuntu.com/ubuntu jammy-security/main i386 Packages [402 kB]
Des:10 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [212 kB]
Des:11 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [1 387 kB]
Des:12 http://mx.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [569 kB]
Des:13 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [228 kB]
Des:14 http://security.ubuntu.com/ubuntu jammy-security/universe i386 Packages [590 kB]
Des:15 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [839 kB]
Des:16 http://mx.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [273 kB]
Des:17 http://mx.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [1 421 kB]
Des:18 http://mx.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [234 kB]
Des:19 http://mx.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1 046 kB]
Des:20 http://mx.archive.ubuntu.com/ubuntu jammy-updates/universe i386 Packages [688 kB]
Des:21 http://mx.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [236 kB]
Descargados 10.9 MB en 3s (3 202 kB/s)
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se pueden actualizar 14 paquetes. Ejecute «apt list --upgradable» para verlos.
```

Figura 1: Impresión de pantalla al ejecutar el comando en la terminal del sistema.

2. El siguiente paso es instalar los paquetes de requisitos previos, para ello se ejecuta en la línea de comandos lo siguiente:

```
sudo apt install \
  apt-transport-https \
  ca-certificates \
  curl \
  software-properties-common
```

Visto desde la terminal:

```

~$ sudo apt install apt-transport-https ca-certificates curl software-properties-common
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
ca-certificates ya está en su versión más reciente (20230311ubuntu0.22.04.1).
fijado ca-certificates como instalado manualmente.
software-properties-common ya está en su versión más reciente (0.99.22.9).
fijado software-properties-common como instalado manualmente.
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi i965-va-driver intel-media-va-driver libaac3 libaom3 libas
libbs2b0 libchromaprint1 libcodec2-1.0 libdav1d5 libflashrom1 libflite1 libftdi1-2 libgme0 libgsm1 libgstre
libnorm1 libopenmpt0 libpgm-5.3-0 libpostproc55 librabbitmq4 librubberband2 libserd-0-0 libshine3 libsnappy
libswscale5 libudfread0 libva-drm2 libva-wayland2 libva-x11-2 libva2 libvdpau1 libvidstab1.1 libx265-199 li
mesa-vdpau-drivers pocketsphinx-en-us va-driver-all vdpau-driver-all
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes NUEVOS:
  apt-transport-https curl
0 actualizados, 2 nuevos se instalarán, 0 para eliminar y 14 no actualizados.
Se necesita descargar 196 kB de archivos.
Se utilizarán 624 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s
Des:1 http://mx.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 apt-transport-https all 2.4.11 [1 510
Des:2 http://mx.archive.ubuntu.com/ubuntu jammy-updates/main amd64 curl amd64 7.81.0-1ubuntu1.15 [194 kB]
Descargados 196 kB en 1s (146 kB/s)
Seleccionando el paquete apt-transport-https previamente no seleccionado.
(Leyendo la base de datos ... 287264 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../apt-transport-https_2.4.11_all.deb ...
Desempaquetando apt-transport-https (2.4.11) ...
Seleccionando el paquete curl previamente no seleccionado.
Preparando para desempaquetar .../curl_7.81.0-1ubuntu1.15_amd64.deb ...
Desempaquetando curl (7.81.0-1ubuntu1.15) ...
Configurando apt-transport-https (2.4.11) ...
Configurando curl (7.81.0-1ubuntu1.15) ...
Procesando disparadores para man-db (2.10.2-1) ...

```

Figura 2: Impresión de pantalla al ejecutar el comando en la terminal del sistema.

APT¹ por defecto utiliza HTTP² para descargar paquetes. Sin embargo, al instalar Docker, se quiere garantizar la seguridad de las transferencias de datos.

apt-transport-https habilita la capacidad de APT para descargar paquetes de forma segura a través de HTTPS³, utilizando el protocolo seguro TLS/SSL⁴.

Por otro lado, ca-certificates, garantiza que el sistema reconozca y confíe en los certificados de las entidades emisoras, validando la autenticidad de los servidores desde donde se descargan los paquetes.

¹APT es el gestor de paquetes de Debian y Ubuntu que simplifica la gestión de software.

²Protocolo estándar para transferir datos en la web. Carece de cifrado, lo que posibilita la interceptación de información por terceros

³Es la versión segura de HTTP, garantiza privacidad y seguridad al descargar paquetes de repositorios en línea.

⁴TLS y SSL cifran la comunicación entre clientes y servidores, proporcionando confidencialidad e integridad a los datos transmitidos.

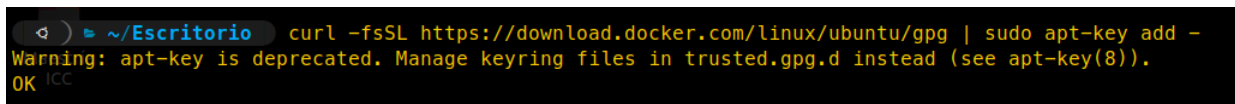
En resumen, al ejecutar la línea de comandos anterior se instalan los paquetes necesarios para habilitar las descargas seguras a través de HTTPS en APT, proporcionando un método seguro para la transferencia de datos durante la gestión de paquetes.

3. Ahora, se añade la clave de GPG para el repositorio de Docker:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Al añadir la clave de GPG (GnuPG Public Key) de Docker, se está asegurando que las descargas provengan de una fuente confiable. La verificación de la firma GPG es una medida de seguridad que garantiza la autenticidad de los paquetes descargados del repositorio de Docker.

Ejecutando desde la línea de comandos se tiene lo siguiente:



```

~ /Escritorio curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK

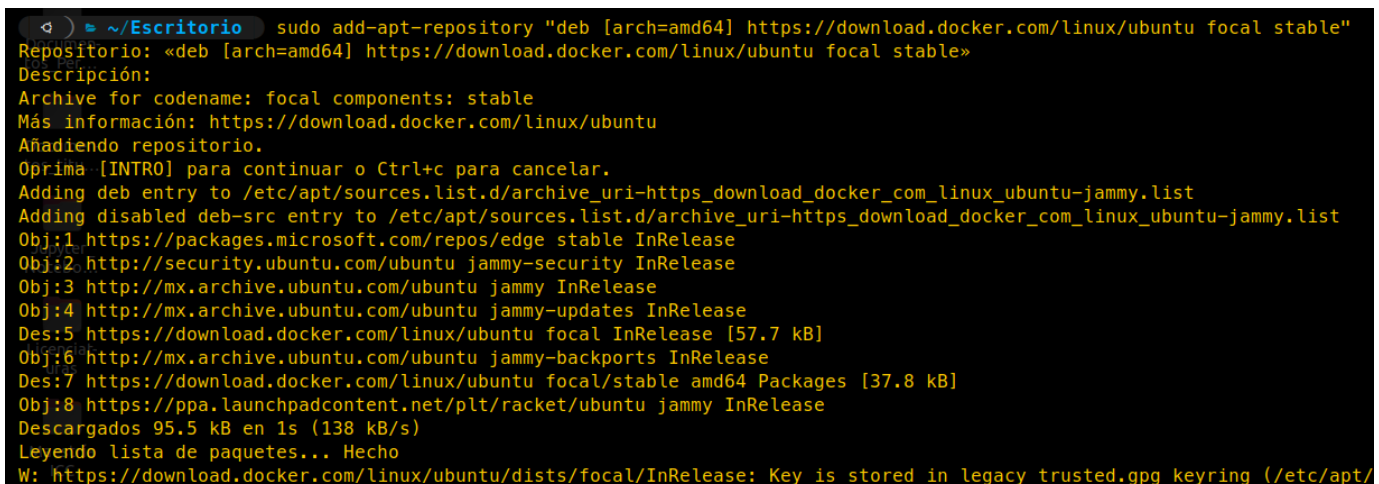
```

Figura 3: Impresión de pantalla al ejecutar el comando en la terminal del sistema.

4. El siguiente paso consiste en agregar el repositorio a las fuentes APT.

```
sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
```

La ejecución desde la terminal muestra lo siguiente:



```

~ /Escritorio sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
Repositorio: «deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable»
Descripción:
Archive for codename: focal components: stable
Más información: https://download.docker.com/linux/ubuntu
Añadiendo repositorio.
Oprima [INTRO] para continuar o Ctrl+c para cancelar.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-jammy.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-jammy.list
Obj:1 https://packages.microsoft.com/repos/edge stable InRelease
Obj:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Obj:3 http://mx.archive.ubuntu.com/ubuntu jammy InRelease
Obj:4 http://mx.archive.ubuntu.com/ubuntu jammy-updates InRelease
Des:5 https://download.docker.com/linux/ubuntu focal InRelease [57.7 kB]
Obj:6 http://mx.archive.ubuntu.com/ubuntu jammy-backports InRelease
Des:7 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages [37.8 kB]
Obj:8 https://ppa.launchpadcontent.net/plt/racket/ubuntu jammy InRelease
Descargados 95.5 kB en 1s (138 kB/s)
Leyendo lista de paquetes... Hecho
W: https://download.docker.com/linux/ubuntu/dists/focal/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/

```

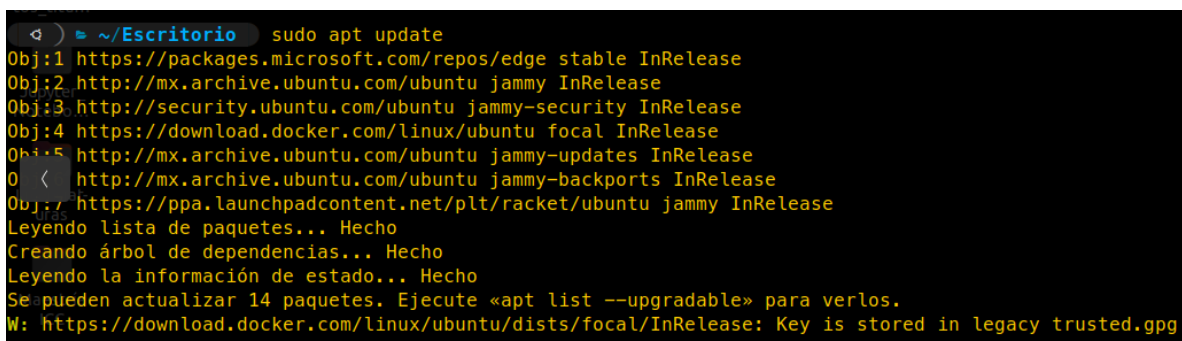
Figura 4: Impresión de pantalla al ejecutar el comando en la terminal del sistema.

Al agregar el repositorio oficial de Docker a las fuentes de APT, se está proporcionando a apt la información necesaria sobre dónde buscar y qué versiones de Docker están disponibles. La inclusión de [arch=amd64] especifica la arquitectura del sistema, y focal es la versión específica de Ubuntu que se está utilizando.

5. Actualizar la base de datos de paquetes

```
sudo apt update
```

Después de agregar el repositorio de **Docker**, es crucial actualizar la base de datos de paquetes para incluir la información más reciente sobre los paquetes disponibles. Esto garantiza que **APT** conozca las últimas versiones y opciones de instalación.



```

~ /Escritorio sudo apt update
Obj:1 https://packages.microsoft.com/repos/edge stable InRelease
Obj:2 http://mx.archive.ubuntu.com/ubuntu jammy InRelease
Obj:3 http://security.ubuntu.com/ubuntu jammy-security InRelease
Obj:4 https://download.docker.com/linux/ubuntu focal InRelease
Obj:5 http://mx.archive.ubuntu.com/ubuntu jammy-updates InRelease
Obj:6 http://mx.archive.ubuntu.com/ubuntu jammy-backports InRelease
Obj:7 https://ppa.launchpadcontent.net/plt/racket/ubuntu jammy InRelease
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se pueden actualizar 14 paquetes. Ejecute «apt list --upgradable» para verlos.
W: https://download.docker.com/linux/ubuntu/dists/focal/InRelease: Key is stored in legacy trusted.gpg

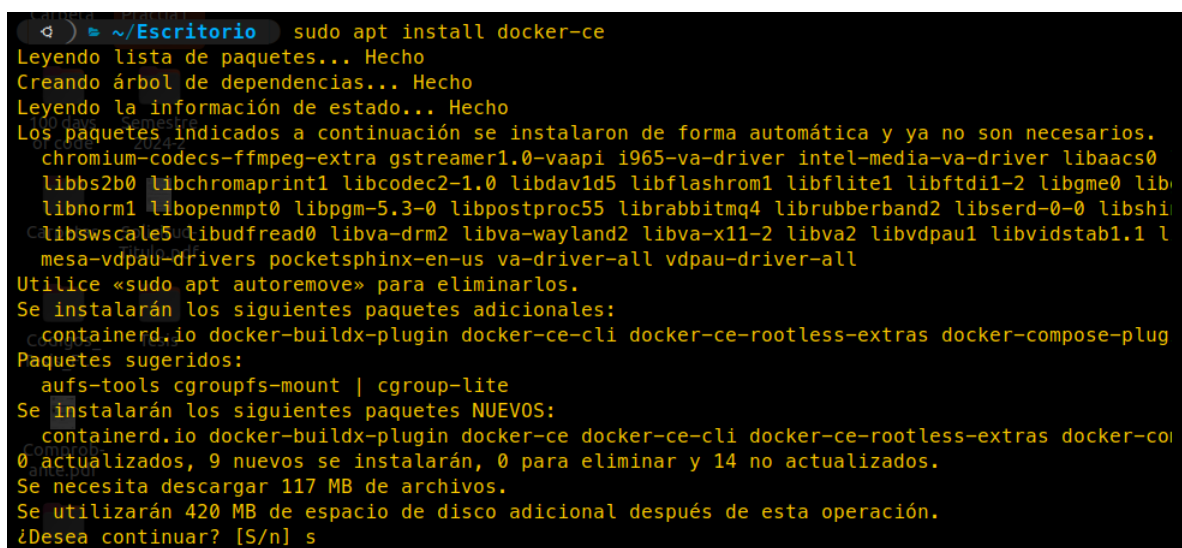
```

Figura 5: Impresión de pantalla al ejecutar el comando en la terminal del sistema.

6. Instalar Docker

```
sudo apt install docker-ce
```

Este paso realiza la instalación real de **Docker** en el sistema. Al utilizar **APT** para gestionar la instalación, se garantiza la coherencia y la integridad de la instalación, así como la facilidad de futuras actualizaciones.



```

~ /Escritorio sudo apt install docker-ce
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi i965-va-driver intel-media-va-driver libaac0
libbs2b0 libchromaprint1 libcodec2-1.0 libdav1d5 libflashrom1 libflite1 libftdi1-2 libgme0 lib
libnorm1 libopenmpt0 libpgm-5.3-0 libpostproc55 librabbitmq4 librubberband2 libserd-0-0 libshi
libswscale5 libudfread0 libva-drm2 libva-wayland2 libva-x11-2 libva2 libvdpau1 libvidstab1.1 l
mesa-va-driver-pocketsphinx-en-us va-driver-all vdpau-driver-all
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
containerd.io docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plug
Paquetes sugeridos:
aufs-tools cgroupfs-mount | cgroup-lite
Se instalarán los siguientes paquetes NUEVOS:
containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-co
0 actualizados, 9 nuevos se instalarán, 0 para eliminar y 14 no actualizados.
Se necesita descargar 117 MB de archivos.
Se utilizarán 420 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s

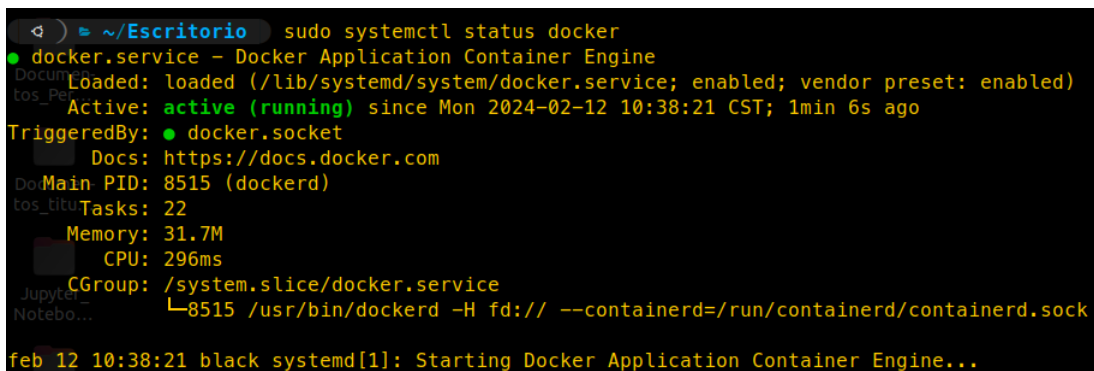
```

Figura 6: Impresión de pantalla al ejecutar el comando en la terminal del sistema.

7. Comprobar el estado del servicio Docker.

```
sudo systemctl status docker
```

Verificar el estado del servicio Docker asegura que el servicio se haya iniciado correctamente. Observar el estado del servicio es esencial para identificar posibles problemas durante la instalación o ejecución.



```
~ /Escritorio sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2024-02-12 10:38:21 CST; 1min 6s ago
     TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 8515 (dockerd)
      Tasks: 22
     Memory: 31.7M
        CPU: 296ms
    CGroup: /system.slice/docker.service
            └─8515 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

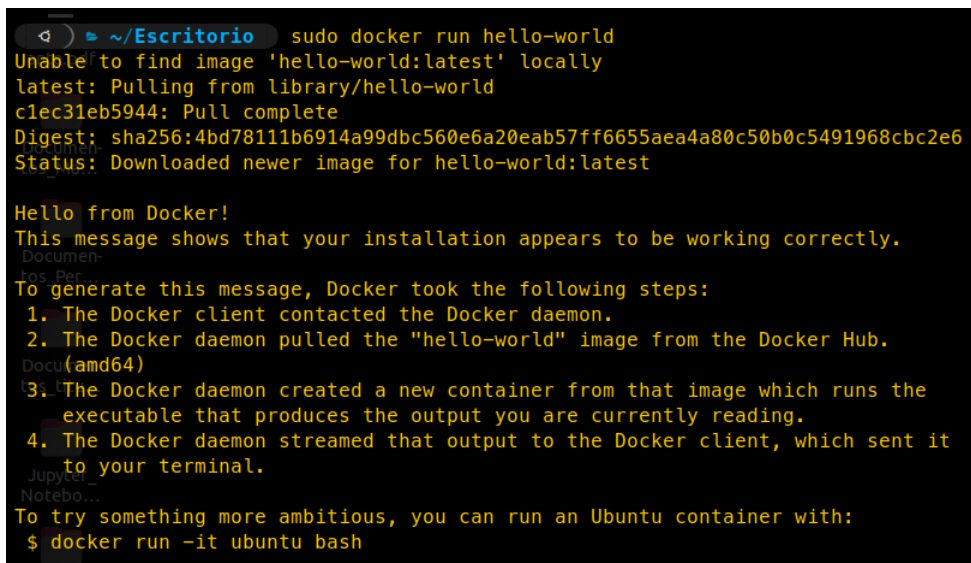
feb 12 10:38:21 black systemd[1]: Starting Docker Application Container Engine...
```

Figura 7: Impresión de pantalla al ejecutar el comando en la terminal del sistema.

8. Probar si se puede trabajar con imágenes de Docker.

```
sudo docker run hello-world
```

Esta prueba simple verifica que Docker puede descargar e iniciar contenedores correctamente. Al ejecutar la imagen "hello-world", se está probando la funcionalidad básica de Docker y asegurándose de que esté listo para su uso.



```
~ /Escritorio sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:4bd78111b6914a99dbc560e6a20eab57ff6655aea4a80c50b0c5491968cbc2e6
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash
```

Figura 8: Impresión de pantalla al ejecutar el comando en la terminal del sistema.

III. INSTALACIÓN DE SMBD POSTGRESQL

Al utilizar contenedores, es posible emplear el SMBD PostgreSQL sin necesidad de una instalación directa. Se siguen los siguientes pasos, enfatizando la importancia de utilizar `sudo` en cada comando:

1. Descargar la imagen de PostgreSQL que Docker utilizará para crear el contenedor:

```
sudo docker pull postgres
```

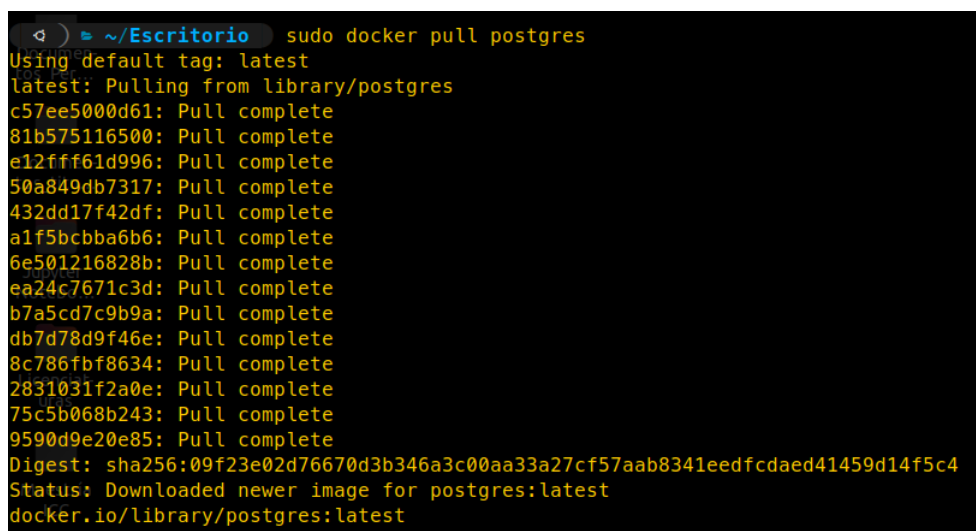


Figura 9: Impresión de pantalla al ejecutar el comando en la terminal del sistema.

Este comando descarga la imagen oficial de PostgreSQL desde el registro de Docker. Descargar la imagen es el primer paso para utilizar PostgreSQL en un contenedor.

2. Una vez descargada la imagen de PostgreSQL, se utiliza el siguiente comando para crear el contenedor:

```
sudo docker run \  
-d \  
--name postgres \  
-e POSTGRES_PASSWORD=mysecretpassword \  
-p 5432:5432 \  
postgres
```

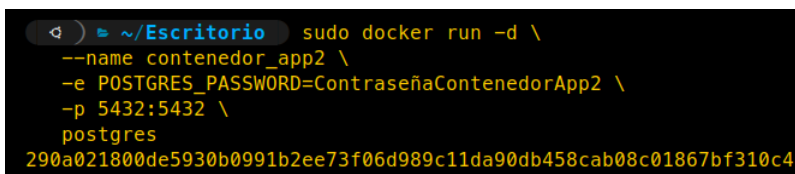
Esta instalación permite ejecutar PostgreSQL de manera aislada en un contenedor, lo que facilita la administración y la portabilidad del entorno de desarrollo.

IMPORTANTE : Consideraciones al crear un contenedor

- a) **Nombre Único:** Cada contenedor debe tener un nombre único en tu sistema. Verifica con el comando `docker ps -a` para evitar conflictos.
- b) **Puertos Únicos:** Asegúrate de que los puertos mapeados desde el contenedor hacia la máquina host sean únicos y no estén en uso por otros procesos o contenedores. Utiliza `sudo lsof -i :<PUERTO>` para verificar.

EJEMPLO : Creación de contenedores

La siguiente captura de pantalla muestra la creación de un nuevo contenedor llamado `contenedor_app2` con contraseña `ContraseñaContenedorApp2` y se ha mapeado el mismo puerto 5432 del contenedor al puerto 5432 del host.

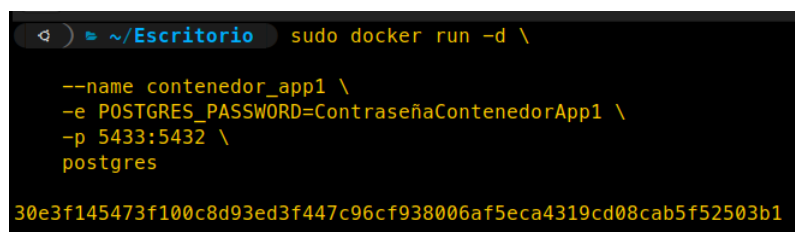


```

< > ~/Escritorio sudo docker run -d \
--name contenedor_app2 \
-e POSTGRES_PASSWORD=ContraseñaContenedorApp2 \
-p 5432:5432 \
postgres
290a021800de5930b0991b2ee73f06d989c11da90db458cab08c01867bf310c4
  
```

Figura 10: Impresión de pantalla al ejecutar el comando en la terminal del sistema.

La siguiente captura de pantalla muestra la creación de un nuevo contenedor llamado `contenedor_app1` con contraseña `ContraseñaContenedorApp1`, se ha mapeado el puerto 5432 del contenedor al puerto 5433 del host para evitar conflictos con otros servicios o contenedores que puedan estar utilizando el puerto 5432.



```

< > ~/Escritorio sudo docker run -d \
--name contenedor_app1 \
-e POSTGRES_PASSWORD=ContraseñaContenedorApp1 \
-p 5432:5433 \
postgres
30e3f145473f100c8d93ed3f447c96cf938006af5eca4319cd08cab5f52503b1
  
```

Figura 11: Impresión de pantalla al ejecutar el comando en la terminal del sistema.

OPCIONAL : Detalles y Consideraciones

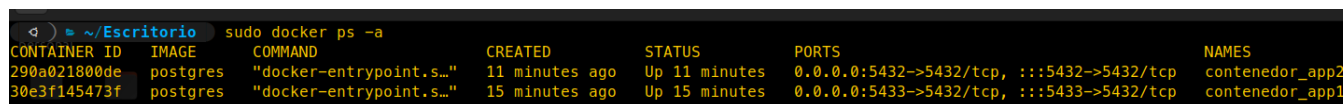
- `docker run`: Crea un contenedor a partir de la imagen Docker.
- `-d`: Ejecuta el contenedor en segundo plano, liberando la terminal para otros usos.
- `-name postgres`: Asigna un nombre al contenedor, facilitando su identificación y gestión.
- `-e POSTGRES_PASSWORD`: Establece la contraseña de PostgreSQL, mejorando la seguridad del sistema.
- `-p 5432:5432`: Mapea los puertos entre la máquina local y el contenedor, permitiendo el acceso a PostgreSQL en el puerto 5432.
- `postgres`: Utiliza la imagen oficial de PostgreSQL sin especificar la versión, asegurando la obtención de la última disponible.

- El puerto estándar 5432 en PostgreSQL facilita la conexión eficiente entre aplicaciones locales y el servidor dentro del contenedor. Aunque se puede optar por otro puerto, usar el estándar simplifica la interoperabilidad y la configuración predeterminada de aplicaciones que se conectan a PostgreSQL.
3. Una vez que el contenedor de PostgreSQL ha sido creado, puede ser necesario realizar operaciones manuales como iniciar, detener o reiniciar el contenedor. Aquí se detallan los pasos y las razones detrás de cada uno:

a) **Verificar el CONTAINER ID:**

```
sudo docker ps -a
```

Este comando lista todos los contenedores, incluso aquellos que no están en ejecución, proporcionando información como el CONTAINER ID, necesario para operaciones específicas.



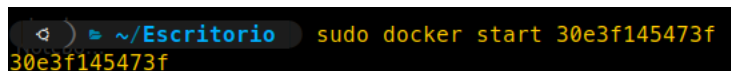
| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |
|--------------|----------|--------------------------|----------------|---------------|---|-----------------|
| 290a021800de | postgres | "docker-entrypoint.s..." | 11 minutes ago | Up 11 minutes | 0.0.0.0:5432->5432/tcp, :::5432->5432/tcp | contenedor_app2 |
| 30e3f145473f | postgres | "docker-entrypoint.s..." | 15 minutes ago | Up 15 minutes | 0.0.0.0:5433->5432/tcp, :::5433->5432/tcp | contenedor_appl |

Figura 12: Impresión de pantalla al ejecutar el comando en la terminal del sistema.

b) **Iniciar el contenedor:**

Tómese por ejemplo el CONTAINER ID del contenedor por nombre `contenedor_app1`.

```
sudo docker start 30e3f145473f
```



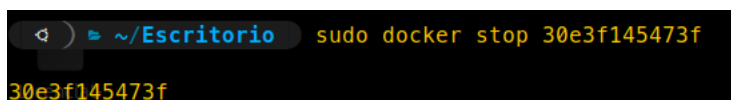
```
sudo docker start 30e3f145473f
30e3f145473f
```

Figura 13: Impresión de pantalla al ejecutar el comando en la terminal del sistema.

Este comando se utiliza para reiniciar un contenedor que ha sido detenido. Es necesario para que PostgreSQL dentro del contenedor vuelva a estar en ejecución.

c) **Detener el contenedor:**

```
sudo docker stop 30e3f145473f
```



```
sudo docker stop 30e3f145473f
30e3f145473f
```

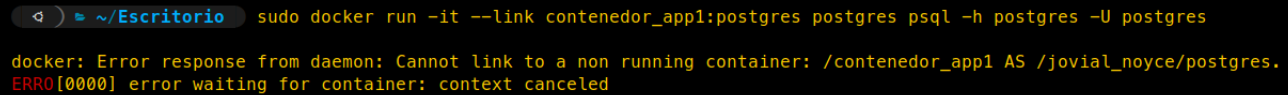
Figura 14: Impresión de pantalla al ejecutar el comando en la terminal del sistema.

Este comando detiene un contenedor en ejecución. Puede ser útil en situaciones específicas, como durante el mantenimiento del sistema o para liberar recursos.

d) Conectar a PostgreSQL desde el contenedor:

```
sudo docker run -it --link postgres:postgres postgres psql -h postgres -U postgres
```

Utilizando el ejemplo anterior, al ejecutar en pantalla, pero con `contenedor_app1` se obtiene el siguiente mensaje en pantalla:



```

~$ sudo docker run -it --link contenedor_app1:postgres postgres psql -h postgres -U postgres
docker: Error response from daemon: Cannot link to a non running container: /contenedor_app1 AS /jovial_noyce/postgres.
ERRO[0000] error waiting for container: context canceled

```

Figura 15: Impresión de pantalla al ejecutar el comando en la terminal del sistema.

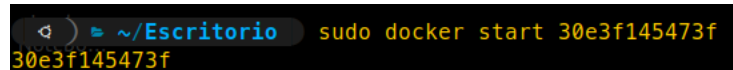
Se obtiene el siguiente mensaje de error:

```

docker: Error response from daemon: \
Cannot link to a non running container: \
/contenedor_app1 AS /jovial_noyce/postgres.
ERRO[0000] error waiting for container: context canceled

```

Este error indica que Docker no puede establecer la vinculación con un contenedor que no está en ejecución (puesto que en c) se detuvo). Para solucionar esto, se debe asegurar de que el contenedor `contenedor_app1` esté en ejecución antes de intentar vincularlo, para ello, se utiliza el comando utilizado en b).



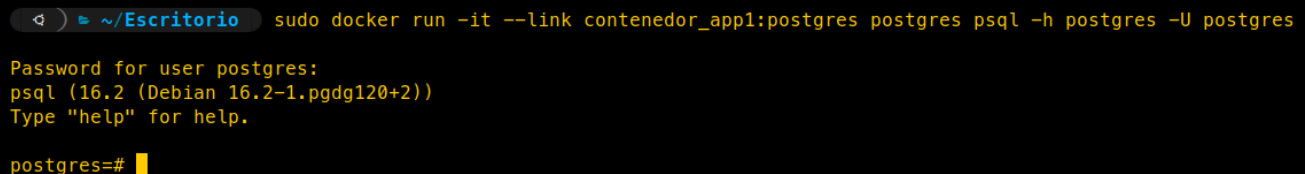
```

~$ sudo docker start 30e3f145473f
30e3f145473f

```

Figura 16: Impresión de pantalla al ejecutar el comando en la terminal del sistema.

Hecho lo anterior, se ejecuta el comando inicial y si todo es correcto, el sistema pide que se ingrese la contraseña del contenedor.



```

~$ sudo docker run -it --link contenedor_app1:postgres postgres psql -h postgres -U postgres
Password for user postgres:
psql (16.2 (Debian 16.2-1.pgdg120+2))
Type "help" for help.

postgres=#

```

Figura 17: Impresión de pantalla al ejecutar el comando en la terminal del sistema.

Este comando permite **interactuar directamente con la consola de comandos de PostgreSQL (psql)** dentro del contenedor. La opción `-link postgres:postgres` vincula el contenedor actual con el contenedor de PostgreSQL para facilitar la conexión.

IMPORTANTE: Para salir de la consola de comandos, se debe oprimir `\q`.

▪ **O P C I O N A L : Explicación Adicional**

- `-it`: Combina las opciones `-i` (interactivo) y `-t` (asigna un terminal pseudo-TTY), permitiendo la interacción con la consola de PostgreSQL.

- `-link postgres:postgres`: Establece un enlace entre el contenedor actual y el contenedor de PostgreSQL, facilitando la conexión.
- `-h postgres -U postgres`: Especifica el host y el usuario para la conexión a PostgreSQL dentro del contenedor.

Estos comandos proporcionan flexibilidad para gestionar y trabajar con el contenedor de PostgreSQL según las necesidades específicas del entorno y del usuario.

PASOS RESUMIDOS

1. Descargar la imagen de PostgreSQL:

```
sudo docker pull postgres
```

2. Crear el contenedor:

```
sudo docker run \  
-d \  
--name postgres \  
-e POSTGRES_PASSWORD=mysecretpassword \  
-p 5432:5432 \  
postgres
```

3. Operaciones con el Contenedor:

■ Verificar CONTAINER ID:

```
sudo docker ps -a
```

■ Iniciar Contenedor:

```
sudo docker start <CONTAINER_ID>
```

■ Detener Contenedor:

```
sudo docker stop <CONTAINER_ID>
```

■ Conectar a PostgreSQL desde el Contenedor:

```
sudo docker run -it --link postgres:postgres postgres psql -h postgres -U postgres
```

INSTALACIÓN DE DBEAVER EN LINUX

1. **Abrir la Terminal:** El usuario debe abrir la terminal en el sistema utilizando el atajo de teclado **Ctrl + Alt + T**.
2. **Ubicarse en el directorio de descarga:** Utilizando el comando **cd**, el usuario debe cambiar al directorio donde descargó el archivo **.deb**. Si el archivo se descargó en el directorio de descargas, el comando podría ser algo así:

```
cd Descargas
```

3. **Instalar DBeaver:** Se debe utilizar el siguiente comando para instalar DBeaver desde el archivo **.deb** que se descargó desde el link <https://dbeaver.io/download/>:

```
sudo dpkg -i dbeaver-ce_23.3.5_amd64.deb
```

En caso de encontrar algún error relacionado con dependencias, el usuario puede ejecutar el siguiente comando para instalarlas automáticamente:

```
sudo apt-get install -f
```

4. **Iniciar DBeaver:** Una vez completada la instalación, el usuario puede iniciar DBeaver desde la terminal o buscarlo en el menú de aplicaciones. Para iniciar DBeaver, se debe escribir el siguiente comando:

```
dbeaver
```

CONEXIÓN A POSTGRESQL DESDE DBEAVER

Se siguen los pasos mostrados en las secciones anteriores para crear un contenedor y conectarse a DBeaver.

1. Descargar la imagen de PostgreSQL:

```
sudo docker pull postgres
```

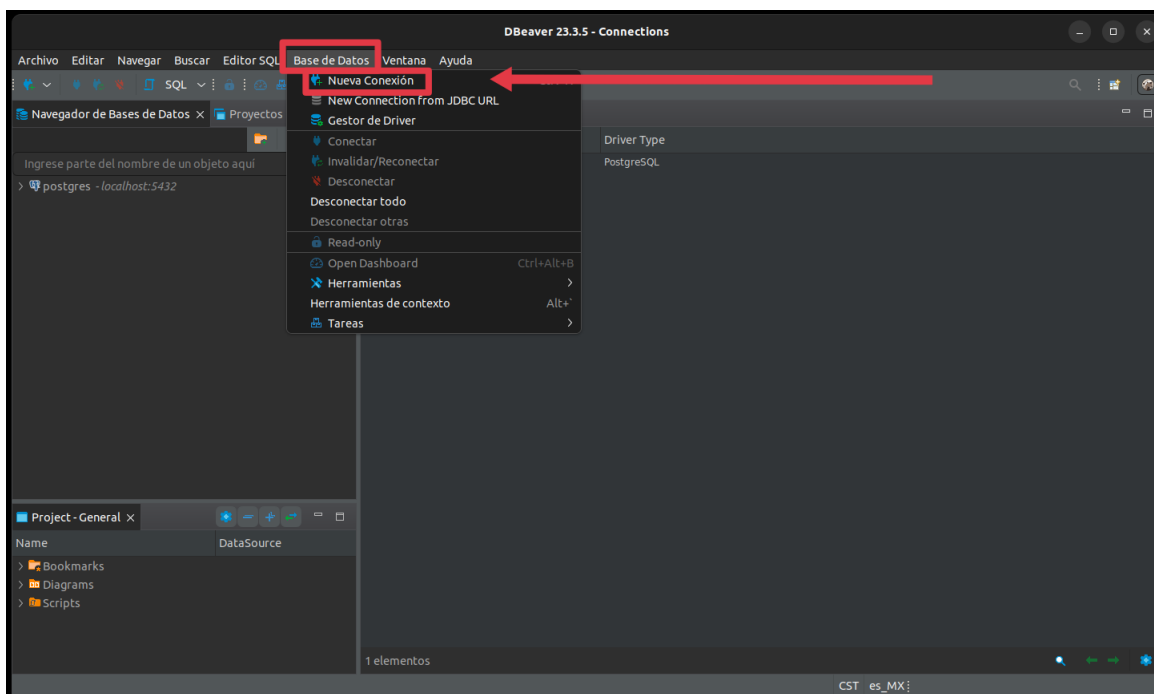
2. Ejecutar el contenedor de PostgreSQL:

```
sudo docker run -d --name prueba_dbeaver \  
-e POSTGRES_PASSWORD=contraseña_dbeaver \  
-p 5432:5432 \  
postgres
```

El comando anterior crea un contenedor de PostgreSQL con el nombre `prueba_dbeaver`, establece la contraseña para el usuario `postgres` como `contraseña_dbeaver` y mapea el puerto 5432 del contenedor al puerto 5432 de la máquina local.

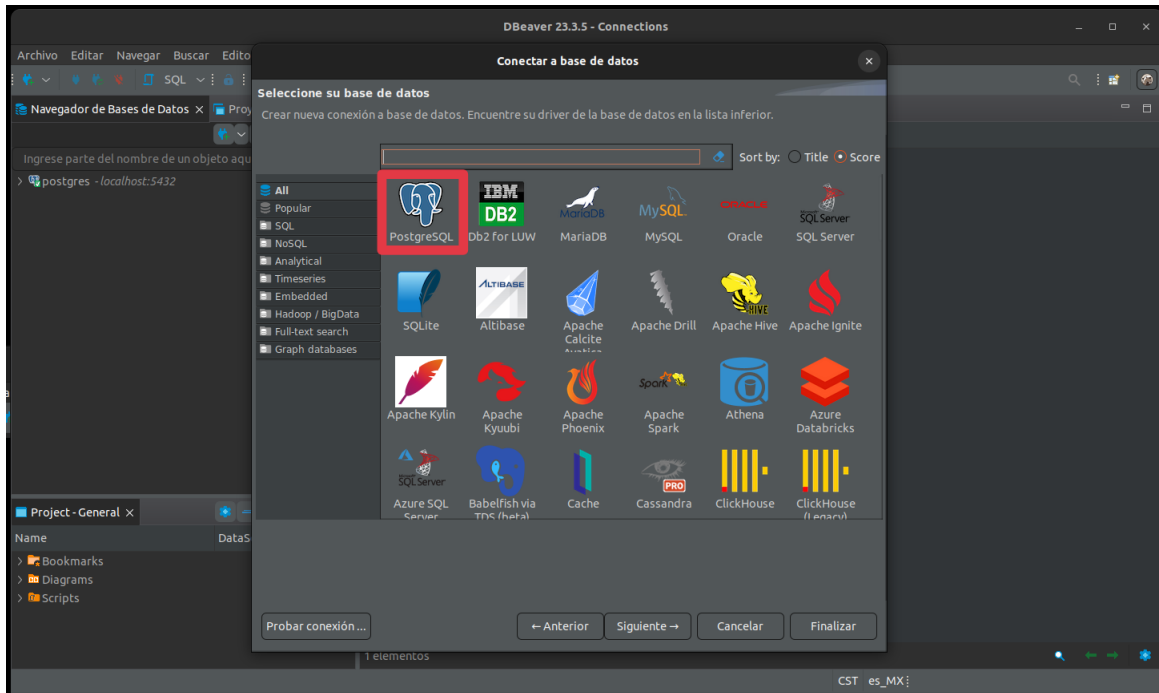
3. Abrir DBeaver: Inicia DBeaver en el sistema.**4. Crear una nueva conexión a PostgreSQL:**

- En la barra de menú de DBeaver, seleccionar Base de Datos y luego Nueva Conexión

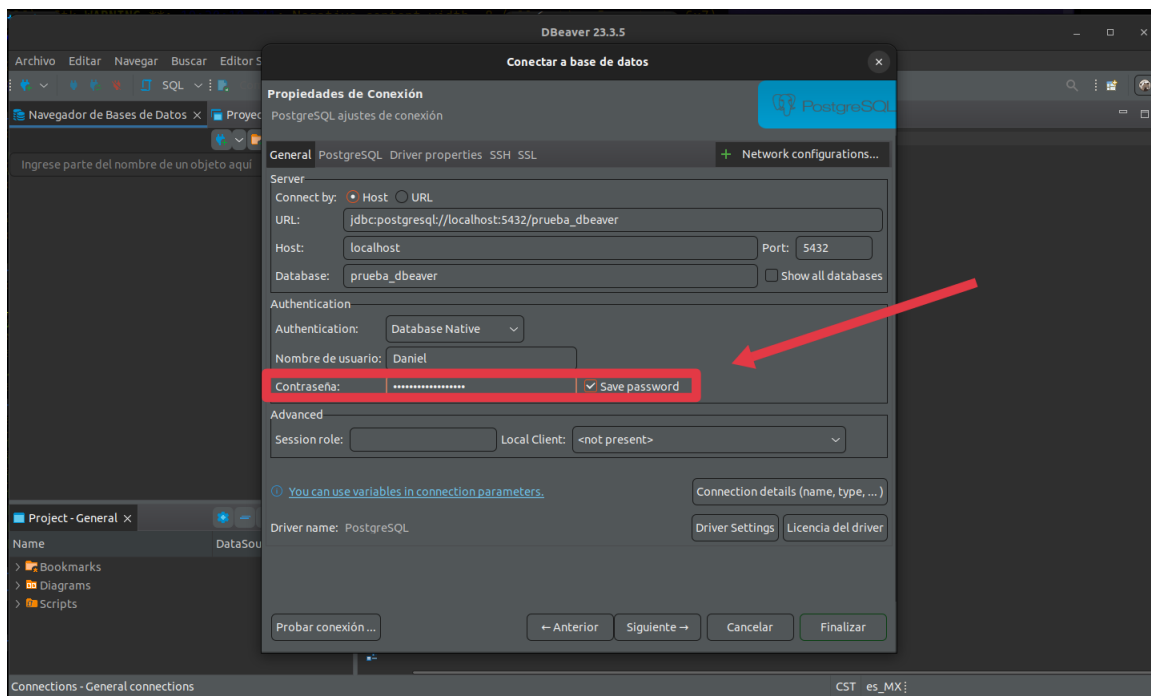


5. Configurar la conexión:

- Seleccionar el tipo de base de datos como PostgreSQL.

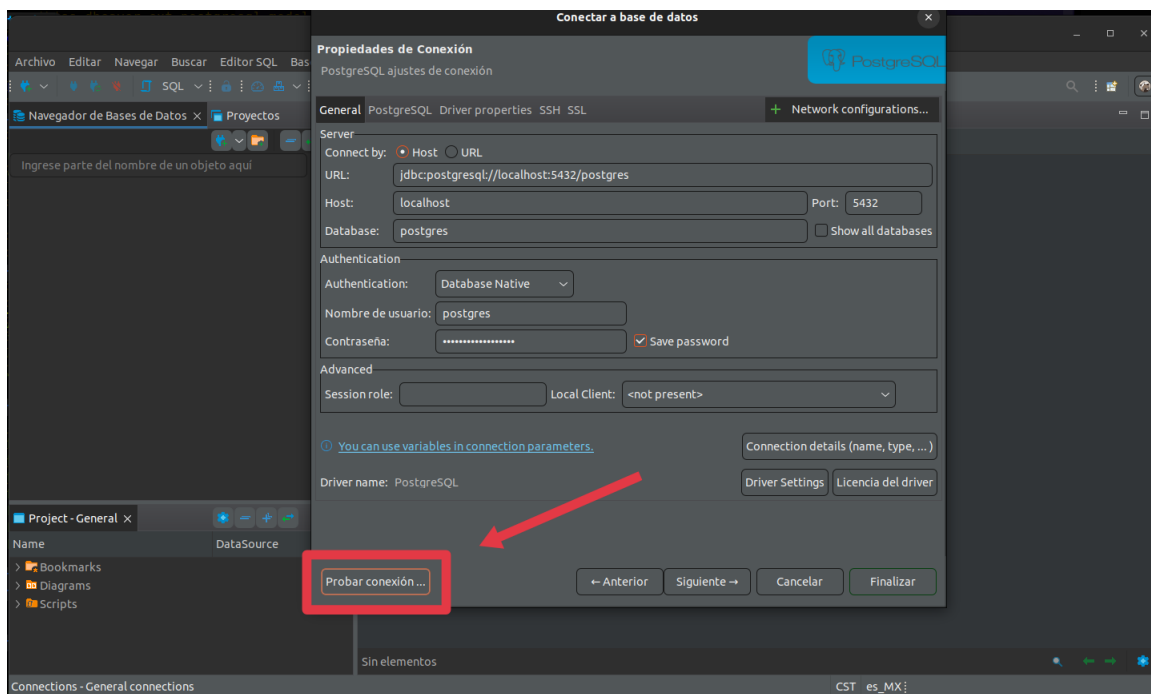


- En la pestaña **Propiedades de conexión**, configurar los siguientes campos:
 - **Host:** localhost (el contenedor se ejecuta en la máquina local).
 - **Port:** 5432 (puerto mapeado desde el contenedor).
 - **Database:** (nombre de la base de datos, por ejemplo, postgres).
 - **Username:** postgres (usuario predeterminado para la imagen de PostgreSQL).
 - **Password:** contraseña_dbeaver (la contraseña establecida al ejecutar el contenedor del paso 2).



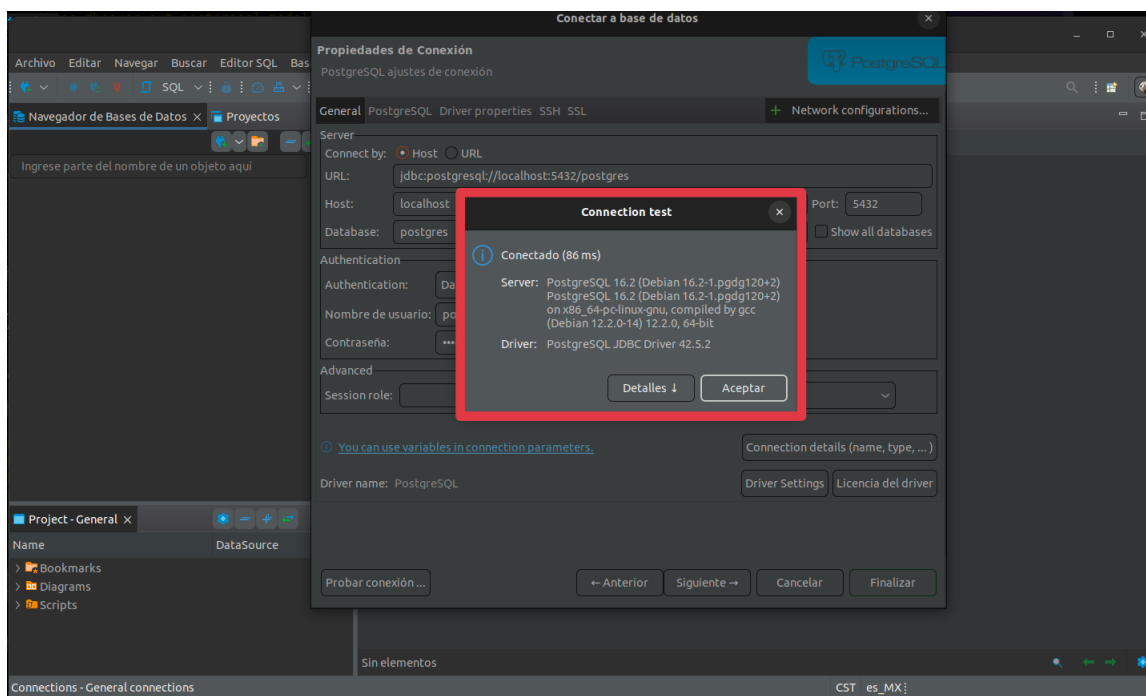
6. Probar la conexión:

- Antes de guardar la conexión, hacer clic en **Probar conexión** para asegurarse de que la configuración es correcta y se puede conectar al contenedor PostgreSQL.



7. Guardar y conectar:

- Si la prueba de conexión es exitosa se mostrará lo siguiente:



- Por último, guardar la configuración y conecta a la base de datos con el botón **Finalizar**.

IV. COMENTARIOS FINALES

La instalación de todo el sistema tomó aproximadamente de 30 a 45 minutos, teniendo en cuenta algunos detalles importantes. Es crucial recordar las contraseñas y utilizar el comando `sudo` antes de los comandos de instalación de PostgreSQL para evitar posibles errores. También, es vital asegurarse de que haya una vinculación con un contenedor en ejecución para evitar problemas.

Un error común es omitir los pasos necesarios:

1. Descargar la imagen de PostgreSQL: `sudo docker pull postgres`
2. Ejecutar el contenedor de PostgreSQL:

```
sudo docker run -d --name prueba_dbeaver
-e POSTGRES_PASSWORD=contraseña_dbeaver
-p 5432:5432
postgres
```

Estos pasos son esenciales para utilizar DBeaver de manera efectiva. Asegurarse de seguir cada paso correctamente facilitará la instalación y evitará errores futuros.