UNIVERSIDAD NACIONAL DEL ALTIPLANO

FACULTAD DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

ESCUELA PROFESIONAL DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

CURSO:
SISTEMAS DISTRIBUIDOS

DOCENTE:
ING. Fred Torres Cruz

PRESENTADO POR:
EDILFONSO MUÑOZ ANCCORI

SEMESTRE: 2024-II

PUNO-PERÚ
2024

Guía Completa para la Configuración de Replicación en PostgreSQL con Docker

December 17, 2024

Abstract

En esta guía, se aborda el proceso completo de configuración de un sistema de replicación de bases de datos utilizando Docker y PostgreSQL. La replicación es una técnica esencial para garantizar la disponibilidad, escalabilidad y resiliencia de las bases de datos en sistemas distribuidos. A lo largo de este documento, se presentan los pasos necesarios para instalar y configurar Docker, así como las mejores prácticas para la creación de contenedores que faciliten la replicación entre instancias de PostgreSQL. Además, se exploran técnicas avanzadas como la configuración de Write-Ahead Logs (WAL), la optimización del almacenamiento mediante Low-Water Marks, y la implementación de Segmentación de Logs para asegurar la integridad y la eficiencia del sistema. Este enfoque garantiza que el entorno de replicación sea altamente disponible y resistente a fallos, adaptándose a diversas necesidades operativas en entornos de producción.

Contents

1 Introducción

La replicación de bases de datos es una técnica crucial en los sistemas distribuidos modernos, ya que garantiza la alta disponibilidad, la tolerancia a fallos y la escalabilidad de los servicios. En sistemas críticos, donde la disponibilidad de los datos es esencial, la replicación se convierte en una estrategia para mantener múltiples copias de la misma base de datos en diferentes ubicaciones, lo que permite que los datos estén siempre accesibles, incluso en caso de fallos en los servidores primarios.

PostgreSQL, uno de los sistemas de gestión de bases de datos relacionales más populares y de código abierto, ofrece capacidades robustas de replicación que permiten distribuir los datos entre diferentes instancias de la base de datos. Estas capacidades incluyen la replicación en tiempo real, garantizando que cualquier cambio realizado en el contenedor principal (líder) se replique automáticamente en las instancias de los contenedores secundarios (seguidores), lo que mejora tanto la resiliencia del sistema como el rendimiento.

El uso de Docker en la implementación de la replicación de PostgreSQL facilita la creación de un entorno controlado y aislado que es fácil de configurar, escalar y administrar. Docker permite la implementación de contenedores ligeros que encapsulan todas las dependencias necesarias, lo que simplifica el proceso de replicación y mejora la portabilidad del sistema.

Esta guía tiene como objetivo proporcionar un proceso paso a paso para configurar un sistema de replicación de PostgreSQL utilizando Docker. A través de este documento, se explorarán desde la instalación de Docker y PostgreSQL, hasta la configuración avanzada de técnicas de replicación como Write-Ahead Log (WAL), archivado de logs, y la optimización del almacenamiento mediante Low-Water Marks. Además, se discutirán buenas prácticas para garantizar que el sistema de replicación sea robusto, escalable y resistente a fallos.

Al final de esta guía, el lector será capaz de configurar un entorno de replicación de PostgreSQL en Docker que pueda ser utilizado en un entorno de producción, mejorando la disponibilidad y fiabilidad de los servicios que dependen de bases de datos críticas.

2 Instalación de Docker y Verificación de Versiones

Docker es una plataforma de contenedores que permite desarrollar, enviar y ejecutar aplicaciones de manera rápida y eficiente. Para empezar a trabajar con Docker y PostgreSQL, primero debemos instalar Docker y Docker Compose en nuestra máquina. A continuación se detallan los pasos para realizar la instalación y verificar que todo esté correctamente configurado.

1. Instalación de Docker

Para instalar Docker en tu sistema operativo, sigue las instrucciones específicas de tu plataforma.

Windows:

1. Visita el sitio web oficial de Docker en `https://www.docker.com/products/docker-desktop`.
2. Descarga el instalador de Docker Desktop para Windows.

3. Ejecuta el instalador y sigue las instrucciones. Es posible que necesites habilitar la virtualización en la BIOS si no está activada por defecto.

4. Una vez completada la instalación, reinicia tu máquina si es necesario.

Mac:

1. Dirígete a `https://www.docker.com/products/docker-desktop` y descarga Docker Desktop para Mac.

2. Abre el archivo descargado y sigue las instrucciones para completar la instalación.

3. Al igual que en Windows, puede ser necesario reiniciar el sistema.

Linux: En distribuciones como Ubuntu, puedes usar los siguientes comandos para instalar Docker:

```
sudo apt update
sudo apt install apt-transport-https ca-certificates curl software-
    ↪ properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
    ↪ add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/
    ↪ linux/ubuntu $(lsb_release -cs) stable"
sudo apt update
sudo apt install docker-ce
```

2. Instalación de Docker Compose

Docker Compose es una herramienta que permite definir y ejecutar aplicaciones multi-contenedor. Para instalar Docker Compose, sigue estos pasos:

Windows y Mac: Docker Compose se instala automáticamente cuando instalas Docker Desktop. No es necesario instalarlo por separado.

Linux: En sistemas basados en Linux, puedes instalar Docker Compose con los siguientes comandos:

```
sudo curl -L "https://github.com/docker/compose/releases/download/$(
    ↪ curl -s https://api.github.com/repos/docker/compose/releases/
    ↪ latest | jq -r .tag_name)/docker-compose-$(uname -s)-$(uname -m)"
    ↪  -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

3. Verificación de Instalación

Una vez que Docker y Docker Compose estén instalados, es importante verificar que ambas herramientas estén funcionando correctamente.

Verificar la instalación de Docker: Abre una terminal o línea de comandos y ejecuta el siguiente comando:

```
docker --version
```

Esto debería devolver la versión instalada de Docker. Un ejemplo de salida podría ser:

```
Docker version 20.10.8, build 3967b7d
```

Verificar la instalación de Docker Compose: De manera similar, verifica la instalación de Docker Compose ejecutando:

```
docker-compose --version
```

Esto debería devolver la versión de Docker Compose, por ejemplo:

```
docker-compose version 1.29.2, build 5becea4c
```

4. Verificar que Docker esté en funcionamiento

Después de instalar Docker, puedes asegurarte de que Docker esté corriendo correctamente ejecutando el siguiente comando:

```
docker run hello-world
```

Este comando descarga una imagen de prueba desde el repositorio de Docker Hub y ejecuta un contenedor. Si Docker está funcionando correctamente, verás un mensaje que confirma que la instalación se ha realizado con éxito.

5. Verificación Visual a través de Docker Desktop (Opcional)

Si estás utilizando Docker Desktop (Windows o Mac), puedes abrir la interfaz gráfica de Docker Desktop para ver los contenedores en ejecución, imágenes descargadas y configuraciones de red, entre otros detalles.

3 Configuración del Proyecto en Docker

Una vez que Docker y Docker Compose están instalados y funcionando correctamente, el siguiente paso es configurar el entorno del proyecto. Esto implica crear un archivo `docker-compose.yml` que definirá los contenedores necesarios para la replicación de PostgreSQL. En este proyecto, se crearán tres contenedores: uno para la base de datos líder (`db_leader`) y dos para las bases de datos seguidoras (`db_follower1` y `db_follower2`).

Estructura del Archivo `docker-compose.yml`

El archivo `docker-compose.yml` es el que define cómo se configuran los contenedores, los volúmenes, las redes y los servicios dentro del entorno de Docker. A continuación se presenta un ejemplo de un archivo `docker-compose.yml` para un entorno de replicación con PostgreSQL:

```
version: '3.8'

services:
  db_leader:
    image: postgres:latest
    environment:
      POSTGRES_USER: Edilfonso
      POSTGRES_PASSWORD: edilfonso2401L
      POSTGRES_DB: database
    volumes:
      - db_leader_data:/var/lib/postgresql/data
    ports:
```

```
      - "5432:5432"

  db_follower:
    image: postgres:latest
    environment:
      POSTGRES_USER: Edil
      POSTGRES_PASSWORD: Edil123
      POSTGRES_DB: database
    depends_on:
      - db_leader
    ports:
      - "5433:5432"

  adminer:
    image: adminer:latest
    ports:
      - "8080:8080"
    volumes:
      - ./custom/custom-style.css:/usr/share/adminer/adminer.css  #
         ↪ Carga el archivo CSS personalizado

volumes:
  db_leader_data:
```

En este archivo:

- **db_leader**: Es el contenedor que actúa como el nodo líder de la replicación. Utiliza la imagen oficial de `postgres:latest`. Se configura con un usuario `POSTGRES_USER`, una contraseña `POSTGRES_PASSWORD` y una base de datos llamada `database`. Los datos de la base de datos se almacenan en un volumen llamado `db_leader_data`.
- **db_follower**: Es el contenedor que actúa como un nodo seguidor de la replicación. También utiliza la imagen de `postgres:latest`. Este contenedor depende del contenedor líder (`db_leader`) y está configurado para escuchar en el puerto 5433.
- **adminer**: Es una herramienta de administración de bases de datos basada en web que se utiliza para gestionar la base de datos PostgreSQL. El contenedor `adminer` se expone en el puerto 8080, y se le asigna un archivo CSS personalizado.
- **volumes**: Se crea un volumen `db_leader_data` para persistir los datos de la base de datos líder. Esto asegura que los datos se mantengan incluso si el contenedor se detiene o elimina.

Configuración de la Red

En este ejemplo, no se ha especificado una red personalizada porque Docker Compose crea una red predeterminada para los contenedores que están en el mismo archivo `docker-compose.yml`. Sin embargo, puedes configurar redes personalizadas si deseas tener más control sobre cómo se comunican los contenedores entre sí.

Configuración de Volúmenes

Los volúmenes se utilizan para almacenar datos persistentes fuera de los contenedores. En este caso, se utiliza un volumen para `db_leader_data`, lo que permite que los datos de la base de datos líder persistan incluso si el contenedor se reinicia o se elimina.

Archivo `docker-compose.yml` Personalizado

Si necesitas personalizar aún más la configuración, como agregar otros contenedores o servicios, puedes modificar el archivo `docker-compose.yml` según sea necesario. Por ejemplo, si quieres agregar un contenedor adicional para realizar respaldos o pruebas, puedes hacerlo fácilmente añadiendo una nueva sección bajo `services`.

Una vez que tengas este archivo `docker-compose.yml` configurado, estarás listo para ejecutar los contenedores de Docker. El siguiente paso es iniciar los contenedores con el comando adecuado.

4 Ejecución de los Contenedores y Verificación

Para ejecutar el archivo `docker-compose.yml`, primero navega a la carpeta del proyecto en la terminal y ejecuta el siguiente comando:

```
cd "D:\UNIVERCIDAD UNAP\CURSOS VII SEMESTRE\SISTEMAS DISTRIBUIDOS\
    ↪ proyecto_docker"
docker-compose up -d
```

El comando `docker-compose up -d` construye y arranca los contenedores en modo `detached`, es decir, en segundo plano. Esto asegura que los contenedores comiencen a ejecutarse sin bloquear la terminal.

Para verificar que los contenedores están en ejecución, puedes usar el comando `docker ps`, que muestra una lista de los contenedores activos. Asegúrate de que los contenedores `db_leader` y `db_follower` estén listados:

```
docker ps
```

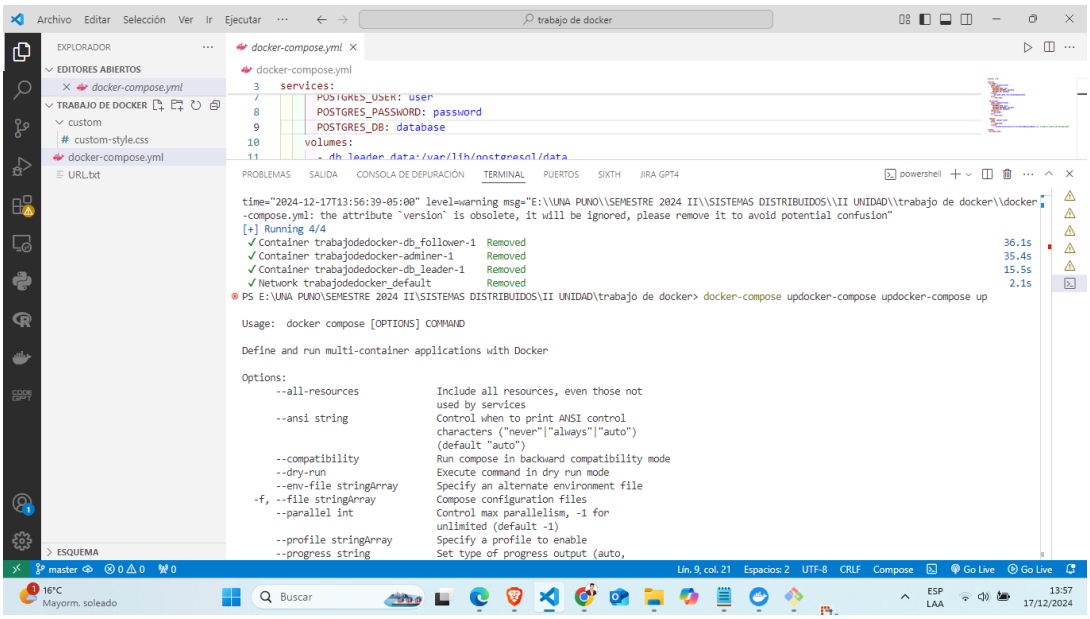

Figure 1: Ejecución de los contenedores con Docker Compose

Esto confirma que los contenedores de PostgreSQL y Adminer están activos y listos para su uso.

5 Configuración de Replicación en PostgreSQL

5.1 Parámetros de Configuración en el Contenedor Líder

La replicación en PostgreSQL depende de algunos parámetros importantes que deben configurarse en el contenedor líder. A continuación, se describen los parámetros clave que deben verificarse:

1. $^{**}wal_level{**}$: $Esteparámetrocontrolalacantidaddeinformaciónqueseescribeenel Write-AheadLog(WAL).Parahabilitarlareplicación, debeestarconfiguradocomo$`replica`.2.$**archive_mode**$: $PermitearchivarlosWALs.Parahabilitarlareplicación, esteparámetrodebeestaractivado$.3.$*wal_keep_size**$: $EspecificaeltamañodelarchivoWALquesemantiene.Ayudaagestionarelalmacenami$

Para verificar la configuración de estos parámetros en el contenedor líder, conéctate al contenedor `db_leader` y ejecuta los siguientes comandos:

```
docker exec -it db_leader psql -U postgres -d main_db
SHOW wal_level;
SHOW archive_mode;
SHOW wal_keep_size;
```

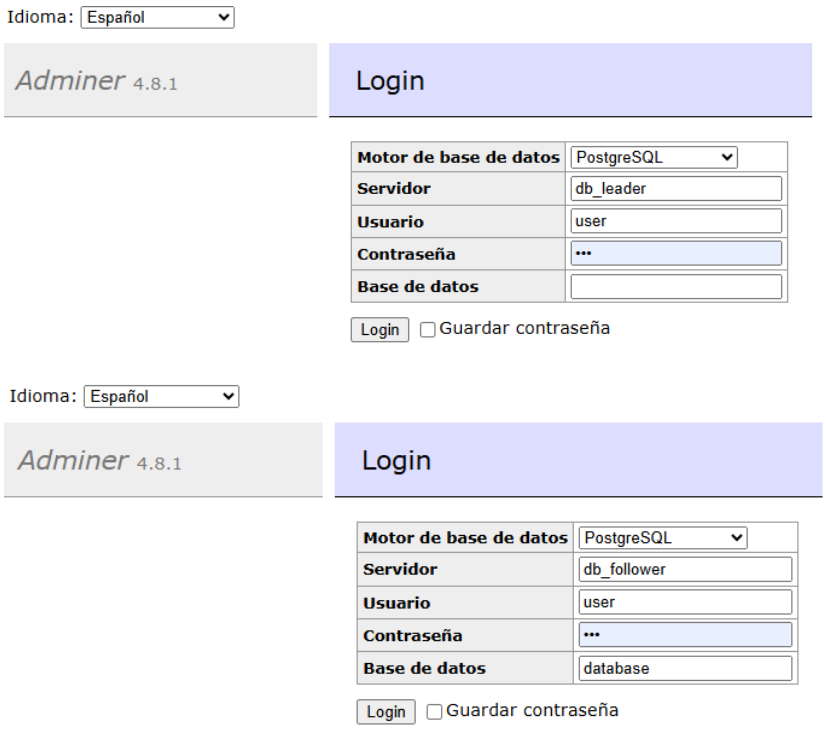

Figure 2: Verificación de parámetros en el contenedor db_leader

Si los valores de los parámetros son correctos (por ejemplo, `wal_level = replica`), entonces la configuración en el contenedor líder está lista para la replicación.

5.2 Verificación en los Seguidores

Una vez que el contenedor líder está configurado, los contenedores seguidores deben recibir los cambios realizados en el líder. Para verificar que los seguidores están recibiendo actu-

alizaciones, conecta con uno de los contenedores seguidores (por ejemplo, `db_follower1`) y ejecuta el siguiente comando:

```
docker exec -it db_follower1 psql -U postgres -d main_db -c "SELECT *
    ↪ FROM pg_stat_replication;"
```

Este comando debe devolver información sobre la replicación en curso, indicando que los seguidores están sincronizados con el líder. La salida debe mostrar los estados de replicación y la conexión activa.

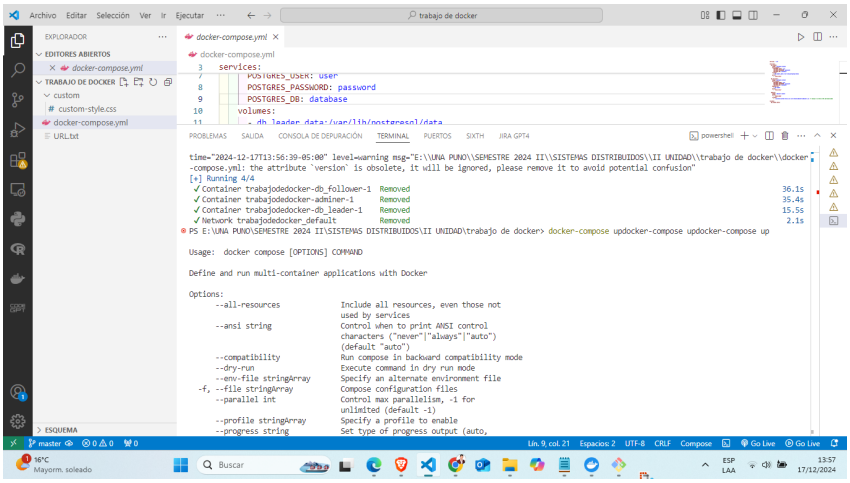

Figure 3: Verificación de la replicación en los contenedores seguidores

6 Técnicas de Replicación

6.1 Write-Ahead Log (WAL)

El `Write-Ahead Log (WAL)` es una técnica utilizada para asegurar que todas las transacciones se registren antes de que se apliquen a la base de datos. Esto es esencial para mantener la integridad de los datos y permitir la recuperación en caso de fallo. En PostgreSQL, el parámetro `wal_level=replica` habilita el registro detallado para permitir la replicación.

- **Configuración**: Asegúrate de que el parámetro `wal_level` esté configurado como `replica` en el contenedor `db_leader`.
- **Código para Verificación en PostgreSQL**:

```
docker exec -it db_leader psql -U postgres -d main_db -c "SHOW
    ↪ wal_level;"
```

- **Resultado Esperado**: El comando debe devolver `replica`, lo que indica que WAL está configurado correctamente para la replicación.

6.2 Segmented Log

La técnica de `Segmented Log` se utiliza para dividir los logs de WAL en segmentos manejables. Esto permite una gestión más eficiente de los logs, especialmente cuando el vol-

umen de datos es grande. La configuración del parámetro `archive_mode=on` habilita el archivado de estos segmentos.

- **Configuración**: Activa `archive_mode=on` en el contenedor `db_leader`.
- **Código para Configuración y Verificación**:

```
# docker-compose.yml
command: >
  -c wal_level=replica
  -c archive_mode=on
  -c archive_command='cp %p /var/lib/postgresql/data/wal_archive/%f
     ↪ '
```

```
docker exec -it db_leader psql -U postgres -d main_db -c "SHOW
   ↪ archive_mode;"
```

- **Resultado Esperado**: El comando debe devolver `on`, indicando que el archivado de segmentos está habilitado.

6.3 Low-Water Mark

La técnica de `Low-Water Mark` ayuda a optimizar el uso del almacenamiento eliminando los logs antiguos que ya no son necesarios para la replicación. Este parámetro se configura con `wal_keep_size`, que define el tamaño de los logs que se deben mantener.

- **Configuración**: Establece el valor de `wal_keep_size` en el contenedor `db_leader` para mantener un tamaño adecuado de los logs antiguos.
- **Código para Configuración**:

```
# docker-compose.yml
command: >
  -c wal_keep_size=128MB
```

- **Verificación**: Ejecuta el siguiente comando para verificar la configuración:

```
docker exec -it db_leader psql -U postgres -d main_db -c "SHOW
   ↪ wal_keep_size;"
```

  Este comando debe devolver el valor configurado de `wal_keep_size`.

7 Recomendaciones

A continuación se presentan algunas recomendaciones clave para optimizar y mantener un sistema de replicación de bases de datos utilizando Docker y PostgreSQL:

- **Monitoreo de los Contenedores**: Es fundamental monitorear los contenedores para garantizar que la replicación funcione correctamente. Utiliza herramientas como `docker stats` para supervisar el rendimiento y el uso de recursos, y `docker logs` para revisar los registros de los contenedores.

- **Respaldo de los Datos**: Aunque la replicación mejora la disponibilidad de los datos, siempre es recomendable realizar copias de seguridad periódicas de las bases de datos. Asegúrate de tener un plan de respaldo que incluya tanto los datos como la configuración de los contenedores y el sistema.

- **Revisar la Configuración de Red**: En entornos con múltiples contenedores o nodos, es importante asegurarse de que la red esté correctamente configurada para permitir la comunicación eficiente entre el contenedor líder y los seguidores. Asegúrate de que los puertos y redes virtuales estén adecuadamente definidos en el archivo `docker-compose.yml`.

- **Manejo de Fallos y Recuperación**: Para un sistema de replicación resiliente, implementa procedimientos de recuperación ante fallos. Por ejemplo, si el contenedor líder falla, asegúrate de que los contenedores seguidores puedan convertirse en el nuevo líder sin pérdida de datos.

- **Uso de Volúmenes Persistentes**: Utiliza volúmenes persistentes en Docker para almacenar los datos de la base de datos. Esto garantiza que los datos no se pierdan si los contenedores se detienen o se reinician, lo que es especialmente importante para entornos de producción.

- **Optimización del Rendimiento de la Replicación**: Ajusta los parámetros de configuración de PostgreSQL, como `max_replication_slots` y `max_wal_senders`, según el número de seguidores y la carga de trabajo esperada. Esto puede mejorar el rendimiento de la replicación y evitar cuellos de botella.

- **Documentación Clara y Completa**: Asegúrate de documentar claramente el proceso de configuración y los procedimientos operativos. Una buena documentación facilitará el mantenimiento y la resolución de problemas a largo plazo.

- **Actualizaciones Regulares**: Mantén tus contenedores y las imágenes de Docker actualizadas. Las nuevas versiones de PostgreSQL y Docker a menudo incluyen mejoras de seguridad, rendimiento y nuevas características que pueden beneficiarte.

Conclusión: Siguiendo estas recomendaciones, podrás garantizar que tu sistema de replicación de bases de datos en Docker y PostgreSQL sea eficiente, seguro y fácil de mantener. Además, asegúrate de realizar pruebas regulares para detectar posibles problemas y mejorar continuamente la infraestructura.

8 Conclusiones

La replicación de bases de datos en entornos Dockerizados con PostgreSQL ofrece una solución robusta y escalable para garantizar la disponibilidad y la resiliencia de las aplicaciones. A través de la utilización de contenedores, se puede gestionar de manera eficiente la configuración y el despliegue de bases de datos en múltiples instancias, lo que permite mejorar el rendimiento y la redundancia de los sistemas de información.

En este proyecto, hemos demostrado cómo configurar un sistema de replicación utilizando Docker y PostgreSQL. Al establecer un contenedor líder y varios seguidores, se ha logrado un entorno de alta disponibilidad, donde las actualizaciones de la base de datos principal se propagan a los nodos secundarios de manera casi instantánea. Este tipo de

configuración es ideal para entornos de producción donde la disponibilidad continua es crítica.

Algunas de las ventajas clave de utilizar Docker en combinación con PostgreSQL para la replicación incluyen:

- **Escalabilidad**: Docker permite replicar fácilmente la base de datos a medida que se incrementa la carga de trabajo, proporcionando la capacidad de escalar horizontalmente al añadir más contenedores seguidores según sea necesario.
- **Aislamiento de Entornos**: Cada contenedor opera de manera independiente, lo que proporciona aislamiento entre las distintas instancias de la base de datos. Esto no solo mejora la seguridad, sino que también facilita la gestión de recursos y la resolución de problemas.
- **Resiliencia ante Fallos**: La replicación de bases de datos, combinada con el uso de Docker Compose, permite configurar un sistema que puede recuperarse rápidamente ante fallos. En caso de que el contenedor líder falle, uno de los seguidores puede asumir el rol de líder sin pérdida de datos.
- **Portabilidad**: Al utilizar Docker, el sistema de replicación se vuelve completamente portátil, lo que facilita su implementación en diferentes entornos de desarrollo, prueba y producción sin preocupaciones por diferencias en la infraestructura.
- **Mantenimiento Simplificado**: Docker y PostgreSQL permiten un enfoque modular en el que es más fácil gestionar y actualizar los contenedores individualmente, sin afectar la operación general del sistema. Además, el uso de volúmenes persistentes garantiza que los datos sean duraderos, incluso si los contenedores se detienen o se eliminan.

Sin embargo, aunque Docker y PostgreSQL proporcionan una infraestructura poderosa, también existen desafíos a tener en cuenta:

- **Gestión de Recursos**: Los contenedores pueden consumir una cantidad considerable de recursos, especialmente cuando se ejecutan múltiples instancias en un solo servidor. Es crucial monitorear el rendimiento y ajustar la configuración de recursos según sea necesario.
- **Seguridad**: Aunque los contenedores proporcionan aislamiento, la seguridad debe ser una prioridad. Se deben aplicar medidas de seguridad como la configuración adecuada de redes, la gestión de contraseñas y la actualización regular de imágenes de Docker para evitar vulnerabilidades.
- **Complejidad en Grandes Escalas**: En sistemas más grandes, la gestión de replicación y la configuración de múltiples contenedores puede volverse compleja. Es importante contar con herramientas de orquestación como Kubernetes para facilitar la administración de contenedores a gran escala.
- **Consistencia en la Replicación**: Si bien la replicación en PostgreSQL garantiza la alta disponibilidad, es fundamental asegurarse de que los parámetros de configuración (como `wal_level`, `archive_mode` y `wal_keep_size`) estén bien configurados para evitar la pérdida de datos o inconsistencias entre el líder y los seguidores.

En conclusión, Docker y PostgreSQL ofrecen una solución flexible y potente para la replicación de bases de datos, que permite mejorar la disponibilidad, el rendimiento y la escalabilidad de las aplicaciones. Sin embargo, como con cualquier tecnología, es esencial considerar los desafíos asociados con su implementación y mantenimiento. Con las configuraciones adecuadas, esta combinación puede ofrecer un sistema de base de datos altamente eficiente y resiliente que se adapte a las necesidades cambiantes de las aplicaciones modernas.

References

[1] Docker, Inc. *Docker Documentation*
`https://docs.docker.com/`.

[2] PostgreSQL Global Development Group. *PostgreSQL Documentation*
`https://www.postgresql.org/docs/`.

[3] Docker, Inc. *Docker Compose Documentation*
`https://docs.docker.com/compose/`.

[4] PostgreSQL Documentation. *Write-Ahead Logging in PostgreSQL*
`https://www.postgresql.org/docs/current/wal.html`.

[5] The Kubernetes Authors. *Kubernetes Documentation*
`https://kubernetes.io/docs/`.

9 Anexo

El código fuente completo de este trabajo está disponible en el repositorio de GitHub. Puedes acceder a él en el siguiente enlace:

`gitcredential-managerrejecthttps://github.com/edilfon/Gu-a-Completa-para-la-Config`
`git`

En este repositorio, encontrarás todos los archivos relacionados con la configuración de Docker, el proyecto en PostgreSQL, y las instrucciones necesarias para la replicación de bases de datos, junto con ejemplos de uso.