# Configuración Optimizada de PostgreSQL y Linux

## 🔹 1. Aumentar el límite de conexiones en PostgreSQL

Edita el archivo de configuración de PostgreSQL:

sudo nano /etc/postgresql/\*/main/postgresql.conf

Modifica o agrega estas líneas:

max\_connections = 2048  
shared\_buffers = 1GB # Ajustado para un servidor con 4GB de RAM  
work\_mem = 4MB # Ajuste conservador para evitar excesivo uso de RAM  
effective\_cache\_size = 2GB # Para mejorar rendimiento de consultas

## 🔹 2. Aumentar el límite de archivos abiertos en Linux (nofile)

Edita /etc/security/limits.conf:

sudo nano /etc/security/limits.conf

Añade al final:

postgres soft nofile 4096  
postgres hard nofile 4096

Verifica que los límites sean aplicados en common-session:

sudo nano /etc/pam.d/common-session

Si no existe, agrégalo:

session required pam\_limits.so

## 🔹 3. Configurar Systemd para PostgreSQL

Edita el servicio de PostgreSQL:

sudo systemctl edit postgresql

Agrega:

[Service]  
LimitNOFILE=4096

Guarda y aplica los cambios:

sudo systemctl daemon-reexec

## 🔹 4. Ajustar límites del kernel

Edita sysctl.conf para aumentar el número de archivos abiertos globalmente:

sudo nano /etc/sysctl.conf

Añade:

fs.file-max = 32768

Aplica los cambios:

sudo sysctl -p

## 🔹 5. Reiniciar PostgreSQL y verificar cambios

Reinicia PostgreSQL para aplicar los cambios:

sudo systemctl restart postgresql

Verifica que los valores se aplicaron correctamente:

ulimit -n  
cat /proc/$(pgrep -u postgres -o)/limits | grep "open files"  
psql -c "SHOW max\_connections;"

## 📌 Nota Final:

Con estos ajustes, PostgreSQL podrá manejar más conexiones sin colapsar el sistema.  
Monitorea el rendimiento con herramientas como htop, pg\_stat\_activity y iostat.  
Si el problema persiste, considera implementar PgBouncer para gestionar las conexiones de manera más eficiente.