

## Unknown Title

---

ChatGPT●

### Tú dijiste:

Instalación y configuración de Bind9 como servidor caché DNS Práctica 3 UT 2: Servicio de Nombres de Dominio (DNS) SERVICIOS EN RED E INTERNET En esta práctica realizaremos la instalación del servidor DNS Bind9 para que funciones como servidor de Caché . Los siguientes pasos hay que resolverlos en dos plataformas: A. En Contenedor Proxmox LXC B. En contenedor Docker usando la imagen ubuntu/bind9 - Docker Image 1. Configura el servidor Bind9 como un servidor caché DNS. Fíjale el tamaño de caché a 256MB. Configura el S.O. 2. Configura tu cliente para que la resolución DNS la realice mediante tu servidor de caché Bind9. 3. Comprueba el correcto funcionamiento : a. Entrando en la web [www.code.org](http://www.code.org) b. Comprueba realizando consultas de distintos dominios usando los clientes DNS nslookup , host y dig . c. Verifica en el servidor que lo que has consultado aparece en la caché y limpia la misma .

ChatGPT Plus

Perfecto 👍 Vamos a detallar **cómo instalar y configurar BIND9 como servidor caché DNS** paso a paso, tanto en **Proxmox LXC** como en **Docker**, y cómo **verificar su funcionamiento**.

---

## OBJETIVO

Configurar un **servidor DNS caché** usando **BIND9**, con un tamaño de caché de **256 MB**, que resuelva nombres para los clientes de la red.

---

## **PARTE A: En Contenedor Proxmox LXC**

### ♦ **1. Instalación de BIND9**

bash

```
apt update
apt install bind9 bind9-utils -y
```

### ♦ **2. Configurar BIND9 como servidor caché**

Edita el archivo principal:

bash

```
nano /etc/bind/named.conf.options
```

Modifica o añade lo siguiente:

bash

```
options {
    directory "/var/cache/bind";

    recursion yes;                // Permite la recursividad
    allow-query { any; };         // Permite consultas desde cualquier host
    allow-recursion { any; };

    // Tamaño de la caché DNS
    max-cache-size 256M;

    // Servidores raíz (por defecto)
    dnssec-validation auto;

    listen-on { any; };
    listen-on-v6 { any; };
};
```

Guarda y sal (Ctrl + O, Enter, Ctrl + X).

### ♦ **3. Reiniciar el servicio**

bash

```
systemctl restart bind9
systemctl enable bind9
systemctl status bind9
```

---

#### ♦ 4. Configurar el cliente DNS

En el cliente (por ejemplo otro contenedor o tu equipo), edita el fichero de resolución:

bash

```
nano /etc/resolv.conf
```

Y deja solo tu servidor DNS:

bash

```
nameserver <IP_DEL_SERVIDOR_BIND9>
```

*(Ejemplo: nameserver 10.0.3.5)*

---

#### ♦ 5. Comprobar funcionamiento

##### a) Prueba con un dominio web

bash

```
ping www.code.org
```

Debe resolver la IP correctamente.

##### b) Usa herramientas de consulta:

bash

```
nslookup www.google.com
host www.openai.com
dig www.wikipedia.org
```

Fíjate en los tiempos: la primera consulta tarda más, la segunda será inmediata (viene de caché).

---

#### ♦ 6. Verificar caché y vaciarla

Para ver estadísticas y caché:

bash

```
rndc stats
```

El archivo de estadísticas se guarda en:

```
swift
```

```
/var/cache/bind/named.stats
```

Para vaciar la caché:

```
bash
```

```
rndc flush
```

---

## PARTE B: En Contenedor Docker

### ◆ 1. Descargar imagen y crear contenedor

```
bash
```

```
docker pull ubuntu/bind9
```

Crea un contenedor:

```
bash
```

```
docker run -d --name bind9-cache \  
  -p 53:53/tcp -p 53:53/udp \  
  -v /srv/bind:/etc/bind \  
  ubuntu/bind9
```

*(La carpeta /srv/bind contendrá los archivos de configuración locales.)*

---

### ◆ 2. Configurar el servidor caché dentro del contenedor

Accede al contenedor:

```
bash
```

```
docker exec -it bind9-cache bash
```

Instala editores si los necesitas:

```
bash
```

```
apt update && apt install nano -y
```

Edita:

bash

```
nano /etc/bind/named.conf.options
```

Y añade lo mismo que antes:

bash

```
options {  
    directory "/var/cache/bind";  
    recursion yes;  
    allow-query { any; };  
    allow-recursion { any; };  
    max-cache-size 256M;  
    dnssec-validation auto;  
    listen-on { any; };  
    listen-on-v6 { any; };  
};
```

Reinicia dentro del contenedor:

bash

```
service bind9 restart
```

---

### ♦ 3. Pruebas desde el host o cliente

En tu máquina física o contenedor cliente, configura `/etc/resolv.conf` apuntando al IP del contenedor Docker:

bash

```
nameserver <IP_DEL_CONTENEDOR_DOCKER>
```

Y ejecuta las mismas pruebas:

bash

```
dig www.code.org  
host www.wikipedia.org  
nslookup www.google.com
```

Comprueba que las respuestas son rápidas en las segundas consultas (se sirven desde caché).

Para limpiar caché:

bash

```
docker exec bind9-cache rndc flush
```

## ✓ VERIFICACIÓN FINAL

Tarea	Comando	Resultado Esperado
Resolución DNS funciona	ping www.code.org	Devuelve IP
Cacheo activo	dig www.google.com (x2)	Segunda consulta rápida
Caché se puede vaciar	rndc flush	Limpieza sin errores
Servicio activo	systemctl status bind9	"active (running)"

¿Quieres que te prepare un **informe en formato PDF o DOCX** con todos los pasos (para entregar como práctica)? Puedo generártelo con las secciones y comandos listos para enviar.

Tú dijiste:

```
De que servidor DNS pilla la información? "options { //tamano del cache DNS en Megas max-cache-size
256m; //Permitir consultas al cache allow-query-cache { any; }; allow-recursion { any; }; directory
"/var/cache/bind"; // If there is a firewall between you and nameservers you want // to talk to, you may
need to fix the firewall to allow multiple // ports to talk. See http://www.kb.cert.org/vuls/id/800113 // If your
ISP provided one or more IP addresses for stable // nameservers, you probably want to use them as
forwarders. // Uncomment the following block, and insert the addresses replacing // the all-0's
placeholder. // forwarders { // 0.0.0.0; // };
//===== // If
BIND logs error messages about the root key being expired, // you will need to update your keys. See
https://www.isc.org/bind-keys
//===== dnssec-
validation auto; listen-on-v6 { any; }; listen-on { any; }; };"
```

ChatGPT Plus

Buena pregunta 🙌

En la configuración que has mostrado, el servidor **BIND9** actúa como **servidor caché recursivo “puro”**, y **no tiene forwarders configurados**.

Eso significa que **no depende de ningún servidor DNS de tu proveedor (ISP)** ni de otro servidor intermedio.

## 🔍 ¿De dónde obtiene entonces la información?

De los **servidores raíz (root servers)** del sistema DNS mundial 🌐.

♦ **Proceso simplificado:**

1. Un cliente (por ejemplo, tu equipo) pregunta a tu servidor BIND9:

```
¿Cuál es la IP de www.code.org?
```

2. BIND9, al no tener la respuesta en su caché, **empieza la resolución recursiva**:

1. Pregunta a los **root servers** (como a `a.root-servers.net`, `b.root-servers.net`, etc.)

→

“¿Quién tiene información sobre el dominio `.org`?”

2. Los root servers responden con las **direcciones de los servidores TLD `.org`**.

3. BIND9 pregunta entonces a los **servidores del TLD `.org`** →

“¿Quién tiene información sobre `code.org`?”

4. Los servidores de `.org` responden con los **servidores autoritativos de `code.org`**.

5. Finalmente, BIND9 consulta a los **servidores autoritativos de `code.org`**, que le devuelven la IP real.

6. Guarda la respuesta en **caché** durante el tiempo indicado por el **TTL** del registro.

---

## ♦ ¿Dónde están definidos los root servers?

En el archivo:

`arduino`

```
/etc/bind/named.conf.default-zones
```

Dentro de él hay una zona llamada `"."` que incluye el fichero:

`swift`

```
/usr/share/dns/root.hints
```

Ese archivo (`root.hints`) contiene una lista actualizada de los 13 servidores raíz del DNS mundial.

Ejemplo de contenido:

`CSS`

```
.                3600000  IN  NS  a.root-servers.net.
a.root-servers.net. 3600000  IN  A   198.41.0.4
b.root-servers.net. 3600000  IN  A   199.9.14.201
...
```

♦ Si quisieras usar servidores *forwarders* (por ejemplo, los de Google o Cloudflare)

Podrías descomentar y modificar esta parte:

bash

```
forwarders {  
    8.8.8.8;  
    1.1.1.1;  
};
```

👉 En ese caso, **tu servidor BIND9 no haría la resolución recursiva completa**, sino que **reenviaría las consultas** a esos DNS externos, que le devolverían las respuestas ya resueltas.

## ✅ En resumen

Modo de operación	De dónde obtiene la información	Configuración
<b>Recursivo (por defecto)</b>	De los <b>root servers</b> del DNS mundial	No hay forwarders
<b>Reenviador (forwarder)</b>	De <b>otros DNS externos</b> (ej. Google 8.8.8.8)	Se activa forwarders { ... };

¿Quieres que te muestre un ejemplo completo de configuración **con forwarders activados** (por ejemplo, usando los DNS de Cloudflare y Google)?

Tú dijiste:

Si

ChatGPT Plus

Perfecto ✅

Aquí tienes un ejemplo **completo y limpio** del archivo

/etc/bind/named.conf.options configurado como **servidor caché con forwarders**, usando **Cloudflare (1.1.1.1)** y **Google DNS (8.8.8.8)**.

## 🧩 Archivo: /etc/bind/named.conf.options

bash

```
options {  
    // Carpeta donde se guardan archivos temporales y de caché  
    directory "/var/cache/bind";
```



```
// Activar recursión (necesario para servidor caché)
recursion yes;

// Permitir consultas al servidor desde cualquier cliente
allow-query { any; };
allow-recursion { any; };

// Tamaño máximo de la caché (256 MB)
max-cache-size 256M;

// Usar DNS externos (reenviadores) para resolver consultas
forwarders {
    1.1.1.1;    // Cloudflare
    8.8.8.8;    // Google
};

// Si los forwarders fallan, intentar resolución directa
forward only;

// Validación de DNSSEC automática
dnssec-validation auto;

// Escuchar en todas las interfaces IPv4 e IPv6
listen-on { any; };
listen-on-v6 { any; };
};
```

## Explicación rápida

Parámetro	Función
recursion yes;	Permite al servidor resolver nombres recursivamente.
allow-query y allow-recursion	Controlan quién puede usar el servidor como resolutor (en este caso, todos).
max-cache-size 256M;	Limita el uso de memoria para la caché.
forwarders	Define los DNS externos a los que reenviar consultas.
forward only;	Obliga a usar solo esos DNS (no intenta contactar con los root servers).
dnssec-validation auto;	Verifica la integridad y autenticidad de las respuestas DNS.
listen-on { any; };	Hace que escuche en todas las interfaces IPv4.
listen-on-v6 { any; };	Hace lo mismo para IPv6.

## Reiniciar y comprobar

Después de guardar el archivo:

bash

```
systemctl restart bind9
systemctl status bind9
```

Verifica que funciona:

bash

```
dig www.code.org
```

Si ves algo como:

yaml

```
;; SERVER: 192.168.1.2#53(192.168.1.2)
;; Query time: X msec
;; FROM: bind9-cache to SERVER: 1.1.1.1#53
```

➔ Significa que tu servidor BIND9 está **reenviando consultas a Cloudflare o Google**.

¿Quieres que te prepare también la **versión sin forwarders (resolución directa con root servers)** para tener ambos ejemplos juntos en tu práctica?

Tú dijiste:

Dime que significa cada línea y si hay algo que falte y porque: options { //tamano del cache DNS en Megas max-cache-size 256m; //Permitir consultas al cache allow-query-cache { any; }; allow-recursion { any; }; directory "/var/cache/bind"; // If there is a firewall between you and nameservers you want // to talk to, you may need to fix the firewall to allow multiple // ports to talk. See <http://www.kb.cert.org/vuls/id/800113> // If your ISP provided one or more IP addresses for stable // nameservers, you probably want to use them as forwarders. // Uncomment the following block, and insert the addresses replacing // the all-0's placeholder. // forwarders { // 0.0.0.0; // };  
//===== // If BIND logs error messages about the root key being expired, // you will need to update your keys. See <https://www.isc.org/bind-keys>  
//===== dnssec-validation auto; listen-on-v6 { any; }; listen-on { any; }; };

ChatGPT Plus

¡Perfecto! Vamos a analizar **línea por línea** tu configuración options y ver si falta algo o hay algo que conviene ajustar. Te lo explico con detalle:

 **Análisis línea por línea**

bash

```
options {
```

- **Inicio del bloque de configuración global** de BIND. Todas las opciones dentro afectan al comportamiento general del servidor.
- 

bash

```
//tamano del cache DNS en Megas  
max-cache-size 256m;
```

- Define **el tamaño máximo de la caché en memoria**.
  - 256m significa 256 megabytes.
  - Limita cuánta memoria puede usar BIND para almacenar respuestas DNS.
  - ⚠ Nota: está bien definido. Sin esto, la caché podría crecer sin límite en sistemas con mucha RAM.
- 

bash

```
//Permitir consultas al cache  
allow-query-cache { any; };
```

- Permite a **cualquier cliente** consultar la **caché** del servidor.
  - { any; } significa “desde cualquier IP”.
  - ⚠ Seguridad: si tu servidor está expuesto a Internet, conviene restringirlo a tu red local para evitar ser usado para ataques de amplificación DNS.
- 

bash

```
allow-recursion { any; };
```

- Permite a **cualquier cliente** realizar consultas recursivas (resoluciones completas) a tu servidor.
  - ⚠ Igual que antes, conviene restringirlo a clientes confiables (allow-recursion { 192.168.0.0/24; };) para no convertir tu servidor en un “open resolver”.
- 

bash

```
directory "/var/cache/bind";
```

- Define la **carpeta donde BIND guarda sus archivos temporales** y de caché.
  - Correcto y necesario.
- 

bash

```
// If there is a firewall between you and nameservers you want
// to talk to, you may need to fix the firewall to allow multiple
// ports to talk.  See http://www.kb.cert.org/vuls/id/800113
```

- Comentario informativo.
  - Indica que si hay firewall, se deben permitir conexiones UDP/TCP en el puerto 53 y rangos dinámicos para la resolución.
- 

bash

```
// If your ISP provided one or more IP addresses for stable
// nameservers, you probably want to use them as forwarders.
// Uncomment the following block, and insert the addresses replacing
// the all-0's placeholder.
```

```
// forwarders {
//     0.0.0.0;
// };
```

- Bloque de **forwarders** comentado.
  - Si se descomenta y se añaden IPs (por ejemplo 8.8.8.8, 1.1.1.1), BIND **reenviará consultas** a esos servidores en lugar de hacer resolución directa desde root servers.
  - Actualmente está comentado → el servidor hace **resolución recursiva completa usando root servers**.
- 

bash

```
//=====
// If BIND logs error messages about the root key being expired,
// you will need to update your keys.  See https://www.isc.org/bind-keys
//=====
dnssec-validation auto;
```

- Activa la **validación DNSSEC automática**, que verifica la autenticidad de los registros DNS firmados digitalmente.
- Es importante para seguridad, especialmente si quieres evitar ataques de suplantación DNS.

---

bash

```
listen-on-v6 { any; };
```

- Hace que BIND escuche en **todas las interfaces IPv6** para consultas DNS.

bash

```
listen-on { any; };
```

- Hace que BIND escuche en **todas las interfaces IPv4** para consultas DNS.
- Si tu servidor solo tiene red local, esto está bien; si está expuesto a Internet, conviene especificar la interfaz local.

bash

```
};
```

- Cierre del bloque options.

---

## Lo que falta o conviene añadir

### 1. Seguridad:

- Restringir `allow-query-cache` y `allow-recursion` a tu red local:

bash

```
allow-query-cache { 192.168.1.0/24; };  
allow-recursion { 192.168.1.0/24; };
```

- Esto evita que tu servidor sea un **open resolver**.

### 2. Forwarders (opcional):

- Si quieres usar DNS públicos como Cloudflare/Google, descomenta y define:

bash

```
forwarders {  
    1.1.1.1;  
    8.8.8.8;  
};  
forward only; // opcional, fuerza a usar solo estos
```

### 3. Logs de consultas (opcional):

- Para depuración o ver la caché, se puede activar logging en `named.conf.local`.

#### 4. Firewall:

- Asegúrate que puertos UDP/TCP 53 estén abiertos en la interfaz que quieras usar.

---

Si quieres, puedo prepararte **una versión final “lista para producción”** de este `named.conf.options` con **seguridad, forwarders y caché a 256MB**, para que la pongas directamente en tu práctica.

¿Quieres que haga eso?

Tú dijiste:

Debería poner algo de esto? "options { directory "/var/cache/bind"; version "No disponible"; listen-on port 53 { any; }; # valor por defecto, pero es mejor ponerlo para saber que lo hemos tenido en cuenta recursion yes; # valor por defecto, pero es mejor ponerlo para saber que lo hemos tenido en cuenta allow-recursion { localnets; }; # valor por defecto, pero es mejor ponerlo para saber que lo hemos tenido en cuenta allow-transfer { none; }; };"


ChatGPT Plus

Sí, **definitivamente conviene incluir varias de esas opciones**, sobre todo si quieres que tu práctica tenga buena **seguridad y claridad**, y además refleje que controlas el comportamiento del servidor. Vamos línea por línea:

---

bash


```
directory "/var/cache/bind";
```

-  Ya lo tenías. Indica dónde BIND guarda archivos temporales y caché. Es obligatorio.

---

bash

```
version "No disponible";
```

-  Opcional pero **muy recomendable**.
- Oculta la versión de BIND que usas. Esto **evita que atacantes sepan tu versión exacta** y pueda ser vulnerable.
- Buena práctica de seguridad en producción.

---

bash

```
listen-on port 53 { any; };
```

- Escucha en todas las interfaces IPv4 en el puerto 53.
  - Aunque `listen-on { any; };` hace lo mismo, **es bueno especificarlo explícitamente** para mostrar que sabes qué puerto y interfaz estás usando.
- 


bash

```
recursion yes;
```

- Ya lo tenías. Permite que BIND haga consultas recursivas.
  - Aunque es el valor por defecto, **es buena práctica ponerlo explícitamente** en la configuración educativa/práctica.
- 

bash

```
allow-recursion { localnets; };
```

- Muy importante .
  - `localnets` permite recursión **solo a redes locales**, evitando que tu servidor se convierta en **open resolver**.
  - Es mucho más seguro que `allow-recursion { any; };` que permitía a cualquiera usar tu servidor para amplificación DNS.
- 

bash

```
allow-transfer { none; };
```

- Muy recomendable.
  - Evita que otros servidores puedan **copiar tu zona DNS**.
  - En un servidor caché no es estrictamente necesario, pero es buena práctica tenerlo.
- 

## Conclusión

Sí, deberías incluir al menos:

1. `version "No disponible";` → seguridad
2. `listen-on port 53 { any; };` → claridad
3. `allow-recursion { localnets; };` → seguridad

4. allow-transfer { none; }; → buena práctica

El resto (directory, recursion yes) ya lo tenías.

---

Si quieres, puedo **escribir tu named.conf.options final combinando tu configuración original con estas mejoras de seguridad y buenas prácticas**, listo para poner en tu práctica y que funcione como servidor **caché seguro**.

¿Quieres que haga eso?

Tú dijiste:

y como se que funciona mi configuración?

ChatGPT Plus

Excelente pregunta 🙌 — saber **si tu configuración de BIND9 funciona correctamente** es fundamental.

Te explico cómo **comprobarlo paso a paso**, desde verificar que el servicio está activo hasta confirmar que la caché funciona.

---

## 1 Verifica que BIND9 está funcionando

bash

```
systemctl status bind9
```

Deberías ver algo así:

pgsql

```
● bind9.service - BIND Domain Name Server
   Loaded: loaded (/lib/systemd/system/bind9.service; enabled)
   Active: active (running)
```

Si aparece “active (running)”  todo va bien.

Si no, revisa el error con:

bash

```
journalctl -xeu bind9
```

---

## 2 Verifica que escucha en el puerto 53



Comprueba que BIND está escuchando correctamente:

bash

```
sudo ss -tulpn | grep named
```

Salida esperada:

nginx

```
udp    UNCONN 0 0 0.0.0.0:53      0.0.0.0:*    users:(( "named",pid=...,fd=...))
tcp    LISTEN 0 10 0.0.0.0:53      0.0.0.0:*    users:(( "named",pid=...,fd=...))
```

➔ Esto confirma que BIND está atendiendo consultas DNS en los puertos UDP/TCP 53.



### 3 Prueba la resolución DNS local

Prueba si tu servidor resuelve correctamente desde **sí mismo**:

bash

```
dig @localhost www.code.org
```

Salida esperada (resumen):

less

```
;; ANSWER SECTION:
www.code.org.    300 IN A 52.84.246.99
```

Si ves una IP en la **ANSWER SECTION**, tu servidor **resuelve correctamente** ✓.



### 4 Prueba desde un cliente de tu red

En otro equipo o contenedor, cambia el DNS en `/etc/resolv.conf`:

bash

```
nameserver <IP_DEL_SERVIDOR_BIND9>
```

Y prueba:

bash

```
ping www.code.org
```

Si responde con la IP, la resolución funciona.

También puedes probar con:

bash

```
nslookup www.google.com
host www.wikipedia.org
dig www.iesflorenciopintado.es
```

---

## 5 Verifica que usa la caché

Haz una consulta dos veces seguidas:

bash

```
dig @localhost www.code.org
dig @localhost www.code.org
```

👉 Si en la segunda vez el **“Query time”** baja mucho (por ejemplo, de 80 ms a 0 ms), significa que la respuesta **vino de la caché** 🟢.

---

## 6 Revisa estadísticas del servidor

Genera estadísticas:

bash

```
rndc stats
```

Esto crea un archivo:

swift

```
/var/cache/bind/named.stats
```

Para leerlo:

bash

```
cat /var/cache/bind/named.stats | grep QUERY
```

Verás las consultas realizadas y cómo BIND las está gestionando.

---

## 7 Limpia la caché y comprueba

Para vaciar la caché:

bash

```
rndc flush
```

Haz otra vez `dig www.code.org` → verás que tarda más la primera vez (porque ha resuelto de nuevo).

---



8

## Revisa los logs

Los logs de BIND suelen estar en:

bash

```
/var/log/syslog
```

Filtra solo los mensajes de named:

bash

```
tail -f /var/log/syslog | grep named
```

Verás consultas como:

CSS

```
named[1234]: client @0x7f12345 192.168.1.5#54535 (www.google.com): query  
(cache) 'www.google.com/A/IN'...
```



## Si todo está correcto, tu servidor:

- Responde consultas (`dig`, `nslookup`, `host` funcionan).
- Guarda en caché (segunda consulta más rápida).
- Muestra actividad en logs.
- Está activo en `systemctl status`.

---

¿Quieres que te dé una **lista corta de comandos “de comprobación rápida”** (por ejemplo, 5 comandos con los que en 30 segundos ves si todo va bien)? Es ideal para incluir al final de la práctica como “verificación final”.

Tú dijiste:


si

Perfecto 😎

Aquí tienes una **lista corta de comprobación rápida** para verificar en **menos de 30 segundos** si tu servidor **BIND9 caché DNS** funciona correctamente.



## Comprobación rápida del servidor BIND9 (modo resumen)

Nº	Comando	Qué comprueba	Resultado esperado
1	<code>systemctl status bind9</code>	Que el servicio está activo	Debe mostrar: active (running) 
2	<code>`ss -tulpn`</code>	<code>grep :53`</code>	Que BIND escucha en el puerto 53
3	<code>dig @localhost www.code.org</code>	Que resuelve dominios	Muestra IP en ANSWER SECTION
4	<code>dig @localhost www.code.org (de nuevo)</code>	Que funciona la <b>caché</b>	“Query time” mucho menor (≈0 ms)
5	<code>rndc flush</code>	Que puedes limpiar la caché	Sin errores → caché vaciada correctamente



### (Opcional) Ver logs en tiempo real:






bash

```
tail -f /var/log/syslog | grep named
```

👉 Te muestra cada consulta DNS que recibe el servidor (útil para ver si los clientes lo están usando).



### Resultado esperado final

-  Servicio activo
-  Escucha en puerto 53
-  Resuelve dominios correctamente
-  Usa caché (segundas consultas más rápidas)
-  Caché se puede vaciar sin errores

¿Quieres que te prepare este resumen en formato **bloque de texto listo para pegar en tu informe o práctica** (con encabezado, comandos y comentarios explicativos)? Quedaría muy bien como “apartado final: Verificación del funcionamiento”.

ChatGPT puede cometer errores. Considera verificar la información importante. Ver [preferencias de cookies](#).