

HOMELAB · PARTE 02 / 02

DESPLEGAR UN RESOLVEDOR DNS DOMÉSTICO

La instalación de AdGuard Home vista como ingeniería de red. Contenedores y publicación de puertos, DNS sobre UDP y TCP, el conflicto con systemd-resolved, cómo DHCP reparte el resolvedor, cifrado hacia fuera y buenas prácticas de seguridad.

Complemento técnico de la guía «Instala AdGuard Home en 4 pasos»

<https://amstech.dev/guias/adguard-home/02-instalacion/>

@ams_tech

ÍNDICE

1	Resumen
2	Arquitectura del despliegue
3	El contenedor
4	DNS sobre UDP y sobre TCP
5	El conflicto del puerto 53
6	El asistente inicial
7	Cómo llega el resolutor a cada dispositivo: DHCP
8	Lo que sale de casa: cifrar y autenticar
9	Seguridad: nunca abrir el 53 a internet
10	Verificación
11	Lo que el vídeo simplifica
12	Referencias

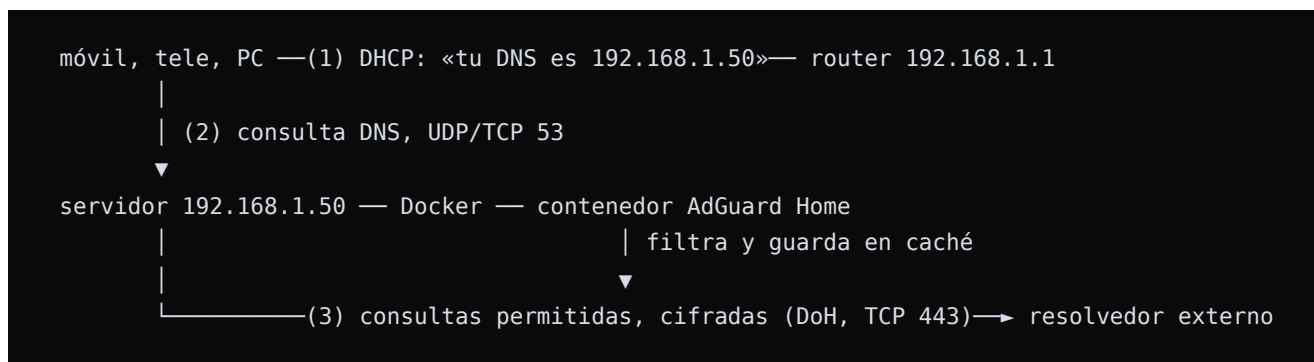
1. RESUMEN

Instalar AdGuard Home son cuatro pasos: elegir dónde, lanzar el contenedor, completar el asistente y anunciarlo en el router. Detrás de cada uno hay una decisión de red. Este dossier explica:

- la arquitectura del despliegue (§2);
- qué hace Docker al publicar puertos y por qué se necesitan volúmenes (§3);
- por qué el DNS necesita a la vez UDP y TCP, y de dónde sale la cifra de 1.232 bytes (§4);
- el conflicto con `systemd-resolved` (§5);
- qué guarda el asistente (§6);
- cómo DHCP entrega el resolutor a cada dispositivo (§7);
- cómo cifrar y autenticar lo que sale de casa (§8);
- por qué el puerto 53 nunca debe quedar abierto a internet (§9);
- cómo verificar el resultado (§10).

Todas las direcciones son de ejemplo: el servidor es `192.168.1.50` y el router, `192.168.1.1`.

2. ARQUITECTURA DEL DESPLIEGUE



Tres flujos: el router **anuncia** el resolutor (1), los dispositivos le **preguntan** (2) y AdGuard Home **reenvía** lo permitido a un resolutor externo (3). El panel web (TCP 3000) es un cuarto flujo, solo de administración.

3. EL CONTENEDOR

3.1 QUÉ ES UN CONTENEDOR

Un contenedor es un proceso normal del sistema al que el núcleo de Linux le da una vista aislada de los recursos: su propio árbol de ficheros, su propia red y su propia lista de procesos. Lo consigue con los **namespaces** del núcleo, y limita lo que puede consumir con los **cgroups**. No hay un sistema operativo virtualizado por debajo, así que arranca en milisegundos y apenas gasta memoria. La **imagen** (`adguard/adguardhome`) es la plantilla de solo lectura de la que sale el contenedor.

3.2 PUBLICACIÓN DE PUERTOS

Por defecto, el contenedor vive en una red virtual interna (`bridge`) con una IP privada que la red de casa no ve. La opción `-p puerto_host:puerto_contenedor/protocolo` pide a Docker que cree reglas de **traducción de direcciones** (DNAT) en el cortafuegos del host [1]: todo lo que llegue a `192.168.1.50:53/udp` se reenvía a la IP interna del contenedor.

OPCIÓN	PROTOCOLO	PARA QUÉ
<code>-p 53:53/udp</code>	UDP	la inmensa mayoría de las consultas DNS
<code>-p 53:53/tcp</code>	TCP	respuestas grandes y reintentos (§4)
<code>-p 3000:3000/tcp</code>	TCP	asistente y panel web

Los puertos por debajo de 1024 son **privilegiados** en Linux: solo `root` (o un proceso con la capacidad `CAP_NET_BIND_SERVICE`) puede escuchar en ellos. Con Docker clásico no es un problema, porque el demonio corre como `root` . En Docker sin privilegios (*rootless*) hay que bajar el umbral con el ajuste del núcleo `net.ipv4.ip_unprivileged_port_start` .

3.3 VOLÚMENES Y PERSISTENCIA

El sistema de ficheros de un contenedor es efímero: al borrarlo, desaparece. Las opciones `-v /opt/adguard/conf:/opt/adguardhome/conf` y `-v .../work:.../work` montan carpetas del host dentro del contenedor [1]. Ahí quedan la configuración (`AdGuardHome.yaml`), las listas descargadas, las estadísticas y el registro de consultas. Así, actualizar consiste en borrar el contenedor y crear otro con la imagen nueva, y el estado sobrevive.

3.4 POLÍTICA DE REINICIO

`--restart unless-stopped` hace que el demonio de Docker vuelva a arrancar el contenedor si el proceso termina o si el host se reinicia, salvo que se haya parado a mano [1]. Para un servicio del que depende toda la red, es el comportamiento correcto: un resolutor caído deja a todos los dispositivos sin poder traducir nombres.

4. DNS SOBRE UDP Y SOBRE TCP

4.1 POR QUÉ UDP

Una consulta cabe en un solo paquete (unos 34 bytes de mensaje, ver el dossier anterior), y UDP la envía sin establecer conexión: un paquete de ida y otro de vuelta. TCP necesitaría primero el saludo de tres pasos (SYN, SYN-ACK, ACK), lo que duplica la latencia sin aportar nada a un intercambio tan corto.

4.2 CUÁNDO TCP

El RFC 1035 limitó los mensajes DNS sobre UDP a **512 bytes** [2]. Si una respuesta no cabe, el servidor la envía recortada con el bit **TC** (*truncated*) activado y el cliente repite la consulta por TCP. El RFC 7766 hizo obligatorio que todo servidor DNS admita TCP [3]. Por eso hay que publicar los dos protocolos: sin TCP, las respuestas grandes (por ejemplo, con firmas DNSSEC o con muchos registros) fallarían.

4.3 EDNS(0) Y EL NÚMERO 1.232

El mecanismo **EDNS(0)** (RFC 6891 [4]) permite al cliente anunciar que acepta respuestas UDP mayores de 512 bytes. ¿Cuánto mayores? Si un paquete UDP supera la **MTU** del camino (el tamaño máximo de paquete que admite cada enlace), se fragmenta, y los fragmentos se pierden o se filtran con frecuencia. El *DNS Flag Day 2020* [5] fijó como valor recomendado **1.232 bytes**, que sale de la MTU mínima garantizada en IPv6:

$$1.280 \text{ (MTU mínima de IPv6)} - 40 \text{ (cabecera IPv6)} - 8 \text{ (cabecera UDP)} = 1.232 \text{ bytes.} \quad (1)$$

Un mensaje DNS de ese tamaño cruza cualquier red IPv6 sin fragmentarse, y por encima se pasa a TCP.

5. EL CONFLICTO DEL PUERTO 53

En muchas distribuciones (Ubuntu entre ellas), `systemd-resolved` ofrece un **resolvidor local** (*stub resolver*) que escucha en `127.0.0.53:53` [6]. Un socket ligado a `0.0.0.0:53` (todas las interfaces) choca con él, y `docker run` falla con «*address already in use*». Para ver quién ocupa el puerto:

```
sudo ss -ltnup 'sport = :53'
```

La solución es desactivar solo ese escucha local, sin desactivar el servicio:

```
sudo sed -i 's/#\?DNSStubListener=.*?/DNSStubListener=no/' /etc/systemd/resolved.conf
sudo systemctl restart systemd-resolved
```

Con `DNSStubListener=no`, el propio servidor seguirá resolviendo nombres a través de los servidores configurados, y el puerto 53 queda libre para AdGuard Home.

6. EL ASISTENTE INICIAL

El asistente (`http://192.168.1.50:3000`) escribe `AdGuardHome.yaml` con tres decisiones:

1. **Interfaz y puerto del panel.** Si se cambia el 3000, hay que publicar el nuevo puerto con otro `-p`.
2. **Interfaz y puerto del DNS.** En un contenedor, «todas las interfaces» se refiere a las del propio contenedor; la exposición real la decide el `-p` del §3.
3. **Credenciales del panel.** La contraseña no se guarda en claro, sino como un hash **bcrypt**: una función deliberadamente lenta y con *sal* aleatoria, diseñada para que adivinar contraseñas a partir del hash sea muy costoso [13].

7. CÓMO LLEGA EL RESOLVEDOR A CADA DISPOSITIVO: DHCP

7.1 EL INTERCAMBIO DORA

Al conectarse a la red, un dispositivo obtiene su configuración por **DHCP** (RFC 2131 [7]) en cuatro mensajes:

```
cliente — DISCOVER (difusión: «¿hay algún servidor DHCP?») —> red
cliente ← OFFER      («te ofrezco 192.168.1.23 y estas opciones») — router
cliente — REQUEST    («acepto 192.168.1.23») —————> router
cliente ← ACK         («confirmado, durante 24 h») ————— router
```

La oferta incluye **opciones** (RFC 2132 [8]). La **opción 3** es la puerta de enlace y la **opción 6**, la lista de servidores DNS. Cambiar en el router el «DNS primario» es cambiar el contenido de la opción 6.

7.2 EL PRÉSTAMO Y SU RENOVACIÓN

La dirección es un **préstamo** (*lease*) con duración. A la mitad del tiempo ($T_1 = 0,5 T$) el cliente intenta renovarlo con el mismo servidor, y al 87,5 % ($T_2 = 0,875 T$) con cualquiera [7]. Por eso un cambio de DNS en el router no llega a todos los dispositivos a la vez: cada uno lo

recibe al renovar, o al reconectarse.

7.3 EL DNS SECUNDARIO

La opción 6 es una lista, pero el estándar no obliga a los clientes a usarla en orden estricto. Algunos sistemas prueban el primero y pasan al segundo solo si no responde. Otros alternan o consultan a varios a la vez y se quedan con la primera respuesta. Si el secundario es un resolvable público, una fracción imprevisible de las consultas se saltará el filtro. Si se quiere redundancia, la opción correcta es una **segunda instancia de AdGuard Home** con la misma configuración, no un resolvable sin filtrar.

8. LO QUE SALE DE CASA: CIFRAR Y AUTENTICAR

Por defecto, las consultas permitidas viajan al *upstream* en claro: el proveedor de internet (o cualquiera en el camino) puede leerlas y alterarlas. Hay dos mejoras independientes:

MECANISMO	QUÉ APORTA	CÓMO
DoH (RFC 8484 [9]) / DoT (RFC 7858 [10])	confidencialidad: nadie en el camino ve qué se consulta	consultas dentro de HTTPS (puerto 443) o TLS (puerto 853) hacia el <i>upstream</i>
DNSSEC (RFC 4033 [11])	autenticidad: la respuesta está firmada por el dueño de la zona	AdGuard Home pide y comprueba las firmas

No son excluyentes: DoH protege el trayecto hasta el *upstream*; DNSSEC protege el dato de extremo a extremo, aunque el *upstream* mintiera. Un *upstream* DoH se configura con su URL, por ejemplo `https://dns.quad9.net/dns-query`.

9. SEGURIDAD: NUNCA ABRIR EL 53 A INTERNET

Un resolvable que responde a cualquiera desde internet es un **resolver abierto**, y se usa en **ataques de amplificación** [12]:

1. El atacante envía consultas pequeñas con la dirección de origen **falsificada**: la de la víctima (UDP no verifica el origen).
2. El resolvable envía respuestas mucho mayores... a la víctima.

El factor de amplificación es tamaño de la respuesta/tamaño de la consulta. Con consultas de unos 60 bytes y respuestas de varios miles de bytes, alcanza decenas de veces. Miles de resolvers abiertos a la vez saturan cualquier conexión.

Reglas prácticas:

- **No redirigir** los puertos 53 ni 3000 en el router. Para usar el filtro fuera de casa, mejor una VPN hacia la red doméstica.

- **Panel con contraseña única.** Si alguien entra, puede ver el historial de consultas de toda la casa.
- **Actualizar la imagen** periódicamente (`docker pull`): un servicio de red es superficie de ataque.

10. VERIFICACIÓN

Desde un ordenador de la red:

```
dig @192.168.1.50 wikipedia.org +short      # debe devolver direcciones
dig @192.168.1.50 api.mixpanel.com +short  # bloqueado: 0.0.0.0
dig @192.168.1.50 wikipedia.org +tcp +short # comprueba que TCP también funciona
```

Para comprobar que DHCP entrega el resolvidor correcto: `resolvectl status` (Linux), `ipconfig /all` (Windows) o los detalles de la red wifi en el móvil. En el panel, el registro de consultas debe mostrar a todos los dispositivos de la casa.

11. LO QUE EL VÍDEO SIMPLIFICA

EN EL VÍDEO	CON PRECISIÓN
«Un solo comando»	Un comando, si Docker ya está instalado y el puerto 53 está libre (§5)
«Abre los puertos»	Crea reglas DNAT del host al contenedor; no abre nada en el router (y no debe, §9)
«Pon esa IP como DNS en tu router»	Cambia la opción 6 de DHCP; cada dispositivo la recibe al renovar su préstamo (§7)
IP del vídeo	<code>192.168.1.50</code> es de ejemplo; no se muestra ninguna IP real

12. REFERENCIAS

- [1] Docker. *Docker Engine: referencia de `docker run`, publicación de puertos, volúmenes y políticas de reinicio*. <https://docs.docker.com/engine/>
- [2] P. Mockapetris. *RFC 1035: Domain Names — Implementation and Specification*. IETF, 1987. <https://www.rfc-editor.org/rfc/rfc1035>
- [3] J. Dickinson et al. *RFC 7766: DNS Transport over TCP — Implementation Requirements*. IETF, 2016. <https://www.rfc-editor.org/rfc/rfc7766>
- [4] J. Damas, M. Graff, P. Vixie. *RFC 6891: Extension Mechanisms for DNS (EDNS(0))*. IETF, 2013. <https://www.rfc-editor.org/rfc/rfc6891>
- [5] DNS Flag Day 2020. *Recomendación de tamaño de búfer EDNS de 1232 bytes*. <https://www.dnsflagday.net/2020/>

- [6] systemd. *resolved.conf(5)* y *systemd-resolved.service(8)*. <https://www.freedesktop.org/software/systemd/man/>
- [7] R. Droms. *RFC 2131: Dynamic Host Configuration Protocol*. IETF, 1997. <https://www.rfc-editor.org/rfc/rfc2131>
- [8] S. Alexander, R. Droms. *RFC 2132: DHCP Options and BOOTP Vendor Extensions*. IETF, 1997. <https://www.rfc-editor.org/rfc/rfc2132>
- [9] P. Hoffman, P. McManus. *RFC 8484: DNS Queries over HTTPS (DoH)*. IETF, 2018. <https://www.rfc-editor.org/rfc/rfc8484>
- [10] Z. Hu et al. *RFC 7858: Specification for DNS over Transport Layer Security (TLS)*. IETF, 2016. <https://www.rfc-editor.org/rfc/rfc7858>
- [11] R. Arends et al. *RFC 4033: DNS Security Introduction and Requirements*. IETF, 2005. <https://www.rfc-editor.org/rfc/rfc4033>
- [12] CISA (US-CERT). *Alert TA13-088A: DNS Amplification Attacks*. 2013. <https://www.cisa.gov/news-events/alerts/2013/03/29/dns-amplification-attacks>
- [13] N. Provos, D. Mazières. «A Future-Adaptable Password Scheme» (bcrypt). *USENIX Annual Technical Conference*, 1999.