Conectividad

- Caddy: Gestión e instalación sencilla de dominios
- Hysteria2: proxy QUIC de alto rendimiento contra bloqueos y restricciones de red
- PiVPN (Wireguard)
- WGDashboard, interfaz web sencilla y funcional para gestionar WireGuard

Caddy: Gestión e instalación sencilla de dominios

Introducción

Este artículo documenta el despliegue y configuración de **Caddy** como reverse proxy principal dentro de la infraestructura. Se utiliza como punto de entrada HTTPS único para servicios internos, encargándose de la terminación TLS, la gestión automática de certificados, el control de acceso mediante autenticación externa y la generación de logs útiles para auditoría.

El enfoque descrito prioriza simplicidad operativa, reducción de errores humanos y una separación clara entre la capa de exposición pública y las aplicaciones internas.

¿Qué es Caddy y por qué se utiliza?

Caddy es un servidor web y reverse proxy moderno, escrito en Go, cuyo diseño gira en torno a HTTPS como comportamiento por defecto. A diferencia de otros servidores tradicionales, la gestión de TLS no es un componente añadido, sino una parte central de su arquitectura.

Sus principales características en este contexto son:

- Gestión automática de certificados TLS mediante ACME.
- Renovación de certificados sin intervención manual.
- Configuración declarativa simple y predecible mediante Caddyfile.
- Recargas en caliente sin interrupción de conexiones.
- Integración nativa con sistemas de autenticación externa mediante `forward_auth`.
- Soporte para validación por DNS challenge, ideal en entornos sin puertos públicos accesibles.

En la práctica, Caddy reduce la complejidad operativa habitual asociada a HTTPS y actúa como frontera clara entre Internet y los servicios internos del stack.

Instalación básica en Debian / Ubuntu

Se recomienda instalar Caddy desde su repositorio oficial para mantener versiones estables y actualizadas:

```
sudo apt update
sudo apt install -y debian-keyring debian-archive-keyring apt-transport-https
curl -fsSL https://dl.caddyserver.com/keys/caddy-stable.asc | sudo gpg --dearmor -o
/usr/share/keyrings/caddy-stable-archive-keyring.gpg
echo "deb [signed-by=/usr/share/keyrings/caddy-stable-archive-keyring.gpg]
https://dl.caddyserver.com/stable/ debian/amd64 stable" | sudo tee
/etc/apt/sources.list.d/caddy-stable.list
sudo apt update
sudo apt install caddy
```

Para otros sistemas o métodos de instalación, se debe consultar la documentación oficial de Caddy.

Puertos y red

Para un funcionamiento estándar con HTTPS automático:

- Puerto **80**: utilizado para validaciones ACME HTTP-01 y redirecciones.
- Puerto **443**: tráfico HTTPS.

Es necesario redirigir ambos puertos al servidor y permitirlos en el firewall:

```
sudo ufw allow 80
sudo ufw allow 443
```

En configuraciones basadas en DNS challenge, el puerto 80 no es estrictamente necesario, pero suele mantenerse por compatibilidad y simplicidad.

Ejemplo básico de Caddyfile

```
wiki.midominio.com {
    reverse_proxy http://192.168.1.10:22300
}

calibre.midominio.com {
    reverse_proxy http://192.168.1.10:20980
```

```
}
```

Aspectos clave:

- Un bloque por dominio.
- `reverse_proxy` redirige el tráfico al servicio interno.
- Los certificados TLS se solicitan y renuevan automáticamente.
- HTTP se redirige a HTTPS sin configuración adicional.

Configuración avanzada con Authentik y Cloudflare

Bloque global

```
{
    acme_dns cloudflare REPLACE_WITH_YOUR_API_TOKEN
    email admin@midominio.com
}
```

Este bloque define opciones globales:

- Uso de DNS challenge mediante Cloudflare para la emisión de certificados.
- El token debe tener permisos mínimos para modificar registros DNS.
- El correo se utiliza para notificaciones del proveedor ACME.

Bloque reutilizable de Authentik

```
(authentik) {
    reverse_proxy /outpost.goauthentik.io/* http://192.168.1.10:15500
    forward_auth http://192.168.1.10:15500 {
        uri /outpost.goauthentik.io/auth/caddy
        copy_headers X-Authentik-Username X-Authentik-Groups X-Authentik-Email X-Authentik-
Name X-Authentik-Uid
    }
}
```

Este bloque define la integración con Authentik:

- El outpost se expone internamente a través de Caddy.

- `forward_auth` valida cada petición antes de llegar al servicio real.
- Los headers copiados permiten identificar usuarios y grupos aguas abajo.

El uso de `import` permite reutilizar esta lógica en múltiples servicios.

Servicios protegidos con logging personalizado

```
wiki.midominio.com {
    import authentik
    reverse_proxy http://192.168.1.10:8080
    log {
        output file /var/log/caddy/wiki-access.log
        format transform "{request>headers>X-Forwarded-For>[0]:request>remote_ip} - {user_id}
[{ts}] \"{request>method} {request>uri} {request>proto}\" {status} {size}" {
            time_format "02/Jan/2006:15:04:05 -0700"
        }
    }
}
```

Este patrón:

- Aplica autenticación antes de acceder al servicio.
- Genera logs con formato estilo nginx.
- Prioriza la IP real del cliente.
- Incluye información del usuario autenticado.

El mismo esquema puede repetirse para otros dominios.

Aplicar cambios

Antes de recargar la configuración:

```
caddy validate --config /etc/caddy/Caddyfile
```

Si no hay errores:

```
sudo systemctl reload caddy
```

La recarga es en caliente y no interrumpe conexiones activas.

Personalización y binarios propios

Para utilizar módulos externos como DNS Cloudflare, puede ser necesario compilar un binario personalizado de Caddy:

- Construcción de Caddy con xcaddy y módulos personalizados

Integración con Authentik:

- Instalación de Authentik
- Proteger aplicaciones web con Authentik

Notas personales

Si Caddy se expone a Internet, es recomendable colocar siempre una capa de autenticación delante de los servicios. La combinación de HTTPS automático, control de acceso centralizado y logging estructurado aporta una base sólida y coherente para exponer aplicaciones de forma segura.

Referencias

- Documentación oficial de Caddy

Hysteria2: proxy QUIC de alto rendimiento contra bloqueos y restricciones de red

Instalación y configuración inicial

Hysteria2, explicado en cristiano: es un protocolo de transporte basado en QUIC que se disfraza muy bien de tráfico legítimo. Su gracia está en que permite saltarse bloqueos, censura y limitaciones de red, porque cifra y mezcla el tráfico de forma que parece otra cosa. A diferencia de un simple proxy, está pensado para rendir en conexiones jodidas (alto lag, pérdidas de paquetes) y para que sea más difícil de detectar. Además, funciona como una VPN de alto rendimiento: cifra todo el tráfico, lo camufla y mantiene velocidades muy altas incluso en redes con restricciones fuertes. Con Blitz, básicamente automatizas su despliegue y gestión, y con Caddy encima lo vistes con certificados TLS de verdad para que no cante a “certificado cutre autofirmado”.

Requisitos previos

- **Sistema**: Debian 11+ / Ubuntu 22+
- **Arquitectura**: x86_64 o ARM64
- **RAM**: 1 GB mínimo
- **Almacenamiento**: 10 GB libres
- **Red**: IPv4/IPv6 compatible
- **Acceso**: privilegios root
- **Dominio**: necesario para el uso de Hysteria2 y compatibilidad con clientes como SingBox
- **Router**: redirección del puerto elegido hacia la IP local del servidor (recomendable usar IP fija)

Instalación base con Blitz

Antes de instalar, actualizar paquetes:

```
apt update && apt upgrade -y
```

Ejecutar el instalador:

```
bash <(curl https://raw.githubusercontent.com/ReturnFI/Blitz/main/install.sh)
```

Al finalizar, se accede al menú principal.

1. Pulsar **1** para entrar en el menú de Hysteria2.
2. Seleccionar **Install and Configure Hysteria2**.
3. Introducir un SNI (o pulsar Enter para usar el predeterminado). Aquí lo correcto es poner el dominio propio que apunte al servidor, ya que el SNI es un nombre de host, no una IP. Si se usa Cloudflare, debe desactivarse para que apunte directamente a la IP real.
   - Ejemplo válido: `bing.com` (sin https:// ni barras). También se puede poner directamente la URL o la IP, siempre que el dominio apunte a la IP del servidor.
4. Elegir un **puerto** de activación (ejemplo: 45443).

> Nota: se recomienda el puerto 443 para mayor ocultación, pero si ya está ocupado (por ejemplo, por Caddy), se puede utilizar otro puerto alternativo. Esto reduce un poco la apariencia de tráfico legítimo, aunque sigue siendo funcional. Algunos clientes permiten usar directamente la IP del servidor como SNI, pero esto no es estándar y puede dar problemas con TLS. Siempre se recomienda usar un dominio con certificado válido (por ejemplo, gestionado por Caddy con Let’s Encrypt).

Uso de certificados con Caddy

Para evitar certificados autofirmados, se puede usar Caddy como emisor y aprovechar los certificados de Let’s Encrypt.

Configuración en `Caddyfile`:

- **Caddyfile en Gitea:** Caddyfile para Hysteria2

Ruta donde Caddy genera certificados (en baremetal):

```
/var/lib/caddy/.local/share/caddy/certificates/acme-v02.api.letsencrypt.org-directory/
```

> **Nota:** el detalle de permisos y acceso a esos certificados se documenta en un artículo aparte: Hysteria2 + Caddy: acceso seguro a certificados (ACL + systemd).

Configuración de Hysteria

El servicio corre bajo hysteria-server.service. Su configuración principal se encuentra en /etc/hysteria/config.json.

Configuración base orientativa disponible en Gitea: config_default.json. Esta configuración es la que genera inicialmente el instalador Blitz, sirve como referencia.

- **Config modificada (certs Caddy, insecure=false, DNS locales, en mi caso, en el tuyo, puedes elegir los que desees, además, los clientes deben estar configurados para usar DNS remoto):** config_modified.json. Esta es la configuración que se recomienda usar, ya que incorpora los ajustes preparados previamente (con los cambios pertinentes, no es para copiar y pegar tal cual, además, debe llamarse config.json).

Flujo de edición rápida

1. Editar `/etc/hysteria/config.json` y apuntar `tls.cert` y `tls.key` a los de Caddy.
2. Revisar permisos y traversal del path como en el artículo dedicado.
3. Reiniciar servicio y revisar `sudo systemctl status hysteria-server.service` para localizar el fallo exacto.
4. Probar conectividad UDP al puerto elegido desde el cliente (por ejemplo con Hiddify, desde Android).

Errores típicos

- `permission denied` al leer `.crt/.key` → revisar artículo de permisos (ACL + systemd).
- `address already in use` → puerto en uso; cambiar `listen` o liberar el socket.
- `tls: handshake failure` → SNI del cliente no coincide, o cert/clave no alinean.
- `no such device` en `bindDevice` → interfaz mal escrita; listar con `ip link`.

Enlaces de interés

- Hysteria2 + Caddy: acceso seguro a certificados (ACL + systemd)
- Documentación de Blitz
- Repositorio Blitz en GitHub
- Hiddify App (Android) → cliente multiplataforma (Sing-box, X-ray, TUIC, Hysteria, Reality, Trojan, SSH, etc.). Desde el panel de Blitz se puede generar un QR para escanearlo directamente con la app y validar el funcionamiento.

Notas

- Este artículo ahora solo mantiene lo básico de instalación y configuración. Los detalles de permisos se han movido a su propio artículo para mantener las cosas ordenadas.

PiVPN (Wireguard)

WireGuard es un protocolo de VPN moderno, rápido y seguro, mientras que **PiVPN** es una herramienta que facilita su instalación y configuración. En este caso, evitamos el asistente automático y usamos una configuración manual para tener más control.

Requisitos previos

- IP fija pública (o DynDNS bien configurado).
- Puerto UDP (por defecto 51820) abierto y redirigido al servidor.
- Sistema basado en Linux con acceso root.

1. Clonar el repositorio de PiVPN

```
git clone https://github.com/pivpn/pivpn.git
cd pivpn
```

2. Preparar la configuración desatendida

```
cd examples
nano unattended_wireguard_example.conf
```

Ejemplo de configuración adaptada:

```
IPv4dev=enp4s0
IPv6dev=enp4s0
install_user=usuario
VPN=wireguard
pivpnNET=10.10.0.0
subnetClass=24
pivpnforceipv6route=1
pivpnforceipv6=0
```

```
pivpnenableipv6=0
pivpnNETv6=fd11:5ee:bad:c0de::
subnetClassv6=64
ALLOWED_IPS="0.0.0.0/0, ::/0"
pivpnMTU=1472
pivpnPORT=51820
pivpnDNS1=8.8.8.8
pivpnPERSISTENTKEEPALIVE=25
UNATTUPG=1
```

---

3. Ejecutar el instalador con esta configuración

```
sudo chmod +x ./auto_install/install.sh
sudo bash ./auto_install/install.sh --unattended ./examples/unattended_wireguard_example.conf
```

---

4. Personalización opcional (clientes sin root)

Editar el archivo para generar perfiles directamente en `/home/usuario/Wireguard`:

```
sudo nano /opt/pivpn/wireguard/makeCONF.sh
```

Ajustar la variable de destino según sea necesario.

---

5. Comandos útiles de PiVPN

```
pivpn -a                  # Añadir nuevo cliente
pivpn -c                  # Ver clientes conectados
pivpn -l                  # Ver todos los clientes
pivpn -qr <cliente>       # QR para importar desde móvil
pivpn -r <cliente>        # Eliminar cliente
pivpn -off / -on <cliente> # Desactivar / activar
```

```
pivpn -d                    # Modo depuración
pivpn -up                   # Actualizar PiVPN
pivpn -bk                   # Backup de la configuración
```

Conclusión

Instalación manual, limpia y personalizable de WireGuard con PiVPN. Esta configuración permite adaptar rutas, IPs, DNS y estructura de archivos sin depender de automatismos cerrados. Ideal para uso doméstico o autoalojamiento avanzado.

WGDashboard, interfaz web sencilla y funcional para gestionar WireGuard

Introducción

WGDashboard es una interfaz web sencilla y funcional para gestionar WireGuard, permitiendo visualizar conexiones, añadir peers y monitorear la actividad en tiempo real. En este artículo, veremos cómo instalarlo en un servidor con Ubuntu 22.04 LTS o 24.02 LTS sin usar Docker, optimizando el rendimiento de nuestra VPN.

Requisitos previos

Antes de comenzar, asegúrate de que tu sistema cuenta con:

- Ubuntu 22.04 LTS o 24.02 LTS
- WireGuard previamente instalado y configurado
- Privilegios de superusuario (`sudo`)
- Una IP fija o servicio DDNS para IPs que no sean fijas.

Si aún no tienes WireGuard instalado, puedes hacerlo con:

```
sudo apt install wireguard -y
```

Instalación paso a paso

Ejecuta el siguiente comando para actualizar paquetes e instalar las herramientas necesarias:

```
sudo apt-get update -y && \
sudo apt install wireguard-tools net-tools --no-install-recommends -y
```

- `wireguard-tools`: Proporciona utilidades esenciales para gestionar WireGuard.
- `net-tools`: Incluye herramientas como `ifconfig` y `netstat`, útiles para depuración y diagnóstico de red.
- `--no-install-recommends`: Evita instalar paquetes adicionales innecesarios.

Clonamos el repositorio oficial de WGDashboard:

```
git clone https://github.com/donaldzou/WGDashboard.git
```

Accedemos al directorio de instalación:

```
cd ./WGDashboard/src
```

Damos permisos de ejecución al script de instalación:

```
chmod +x ./wgd.sh
```

Ejecutamos el instalador de WGDashboard:

```
./wgd.sh install
```

Esto descargará y configurará automáticamente los servicios necesarios para que WGDashboard funcione correctamente.

Habilitar el reenvío de paquetes IPv4

Para que WireGuard funcione correctamente, debemos habilitar el reenvío de paquetes IP:

```
sudo echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf
```

Aplicamos los cambios:

```
sudo sysctl -p /etc/sysctl.conf
```

Configuración de WGDashboard

Para acceder a WGDashboard, asegúrate de que el puerto en el que está configurado (por defecto `10086`) está abierto y redirigido correctamente en tu router hacia la IP del servidor donde está instalado WireGuard.

Abre un navegador e ingresa:

```
http://<IP_DEL_SERVIDOR>:10086
```

Si no conoces la IP de tu servidor, puedes obtenerla con:

```
ip a
```

Configuración de los peers

Dentro de WGDashboard, en la sección **Peer Default Settings**, se configuran los valores por defecto para los nuevos clientes de WireGuard:

- **DNS**: Puedes usar cualquiera. Si tienes Pi-hole o AdGuard Home, pon la IP de tu servidor DNS.
- **Endpoint Allowed IPs**: `0.0.0.0/0`
- **MTU**: `1420`
- **Persistent Keepalive**: `25` (puede depender de si estás detrás de un NAT; en mi caso, lo tengo en `60`).
- **Peer Remote Endpoint**: `TuDirecciónIP:PuertoUsado` (sustituye por tu IP pública y el puerto que redirigiste para WireGuard).

Configuración de la interfaz wg0

Para crear una interfaz de WireGuard (`wg0`), usa los siguientes ajustes:

- **Configuration Name**: `wg0` (puedes cambiarlo, pero es mejor mantenerlo simple).
- **Private Key & Public Key**: Se generan automáticamente.
- **Listen Port**: El puerto que tienes redireccionado en el router para tu servidor WireGuard (por defecto, `51820`).
- **IP Address/CIDR**: `100.10.0.1/24` (puedes modificarlo según tu red).

Opciones avanzadas: Reglas iptables

Para que WGDashboard funcione correctamente, hay cuatro apartados adicionales en la configuración de WireGuard. Asegúrate de aplicar estos cambios:

- **PreUp**: Dejar en blanco.
- **PreDown**: Dejar en blanco.
- **PostUp**:

  ```
  iptables -w -t nat -A POSTROUTING -o enp4s0 -j MASQUERADE
  ```

  En caso de no funcionar, prueba con:

```
iptables -A FORWARD -i wg0 -j ACCEPT; iptables -A FORWARD -o wg0 -j ACCEPT; iptables -t nat -A POSTROUTING -o enp4s0 -j MASQUERADE
```

- **PostDown**:

```
iptables -w -t nat -D POSTROUTING -o enp4s0 -j MASQUERADE
```

En caso de no funcionar, prueba con:

```
iptables -D FORWARD -i wg0 -j ACCEPT; iptables -D FORWARD -o wg0 -j ACCEPT; iptables -t nat -D POSTROUTING -o enp4s0 -j MASQUERADE;
```

Donde `enp4s0` es la interfaz de red principal del servidor. Cámbiala si es diferente en tu caso.

Si no tienes conectividad en LAN, añade esta regla adicional:

```
sudo iptables -t nat -A POSTROUTING -s 100.10.0.0/24 -d 192.168.1.0/24 -j MASQUERADE
```

Guarda los cambios de forma permanente:

```
sudo netfilter-persistent save && sudo netfilter-persistent reload
```

Crear un servicio systemd para WGDashboard

Para que WGDashboard se inicie automáticamente al arrancar el sistema, crearemos un servicio:

Accede a la carpeta `src` de WGDashboard:

```
cd WGDashboard/src
```

Obtén la ruta absoluta del directorio:

```
pwd
```

Edita el archivo `wg-dashboard.service`:

```
nano wg-dashboard.service
```

Sustituye `<absolute_path_of_WGDashboard_src>` por la ruta obtenida. Luego copia el servicio a systemd:

```
sudo cp wg-dashboard.service /etc/systemd/system/wg-dashboard.service
```

Habilita y activa el servicio:

```
sudo chmod 664 /etc/systemd/system/wg-dashboard.service
sudo systemctl daemon-reload
sudo systemctl enable wg-dashboard.service
sudo systemctl start wg-dashboard.service
```

Verifica que está corriendo correctamente:

```
sudo systemctl status wg-dashboard.service
```

Ahora WGDashboard se iniciará automáticamente tras cada reinicio.

Conclusión

Con esta guía, hemos instalado y configurado WGDashboard en Ubuntu, asegurándonos de optimizar su rendimiento sin Docker. Con los ajustes correctos, podemos gestionar nuestra VPN de manera eficiente.

Si necesitas más ajustes o seguridad adicional, revisa la documentación oficial de WGDashboard.