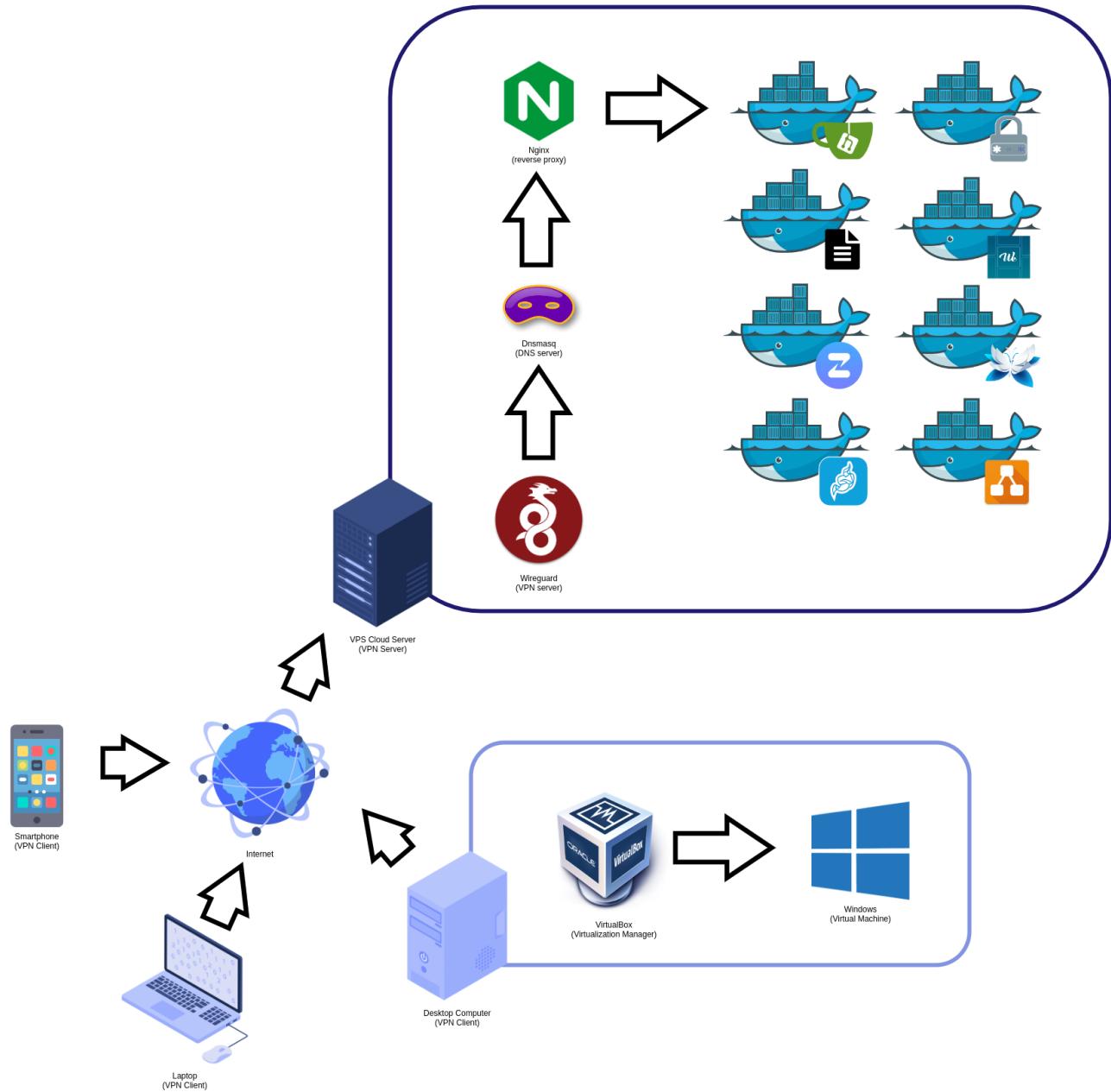


# README



LAB.LOCAL es un ejemplo ficticio del diseño e implementación de la Infraestructura de una organización.

En esta guía se pretende abordar tanto el diseño como las herramientas involucradas en la creación y despliegue de una Infraestructura on-premises utilizando software libre. El propósito es compartir el procedimiento a seguir, así como posibles ideas que adoptar, para toda aquella persona que quiera montar su propio laboratorio u organización.

# Requerimientos

Los requerimientos necesarios para poder seguir esta guía son:

Máquina Virtual - **VPS**:

- Ubuntu Server 20.04 LTS
- 30GB espacio en disco
- 8Gb memoria ram

Máquina Virtual - **LAPTOP**:

- Ubuntu Desktop 20.04 LTS
- 15GB espacio en disco
- 4Gb memoria ram

Opcionalmente, si se puede disponer de un segundo equipo para utilizarlo como "Virtualizador" y conectarlo con la VPN, sería perfecto. No obstante, como se sabe que no es un requerimiento sencillo, simplemente se mostrará el proceso para poder realizar las configuraciones e instalaciones necesarias.

## Note

Este segundo equipo, debe disponer de las características suficientes para poder ejecutar (por lo menos) una máquina virtual además del sistema del host; el cual se recomienda que sea Windows para coincidir con el escenario planteado en esta guía.

## Antes de comenzar...

Pero... antes de comenzar, es importante explicar algunos puntos a tener en cuenta así como esta estructurada la guía.

En primer lugar, la guía se divide en 3 bloques:

1. Instalación y Configuración de los servicios VPN, DNS, Proxy y Contenedores de Aplicaciones.
2. Conexión de "Virtualizador" a la Infraestructura y reglas de enrutado.
3. Estrategia de Backups.

### Note

Para el primer bloque se utilizarán las máquinas virtuales: VPS y LAPTOP. Las cuales configuraremos para poder alojar y acceder a cada uno de los servicios desplegados.

Por otro lado, el segundo bloque, al tener que interconectar una máquina física (Virtualizador), que utilizaremos como equipo que aloja máquinas virtuales; se trasladará la Infraestructura a un VPS en la nube. De esta forma, se podrá conectar y acceder a las máquinas virtuales desde la VPN.

Por último, el tercer bloque, lo realizaremos sobre esta Infraestructura trasladada a la nube, donde se definirá tanto el proceso de backups de contenedores como de las máquinas virtuales del "Virtualizador".

En el primer bloque, para poder realizar las instalaciones necesarias con privilegios de superusuario, se ha creado un usuario administrador local: `localadmin` con password `localadmin`.

### Warning

Se recomienda ir haciendo snapshots de las máquinas virtuales (especialmente VPS), para poder hacer rollback en caso de introducir algún error o para poder volver con facilidad al estado anterior.

### Warning

#### **IMPORTANTE:**

En caso de disponer de un VPS alojado en la nube, que disponga de los requerimientos necesarios que se ha mencionado anteriormente, se recomienda utilizar dicho servidor en lugar de la **máquina virtual: "VPS"**.

Debido a que no todo el mundo dispone de los requerimientos necesarios para montar esta infraestructura (3 máquinas físicas: VPS, LAPTOP y VIRTUALIZADOR), se han creado 2 máquinas virtuales: VPS y LAPTOP, para poder realizar los pasos del primer y tercer bloque.

# Configuración de Red

Antes de comenzar con todo el proceso de instalación y despliegue de la infraestructura, se debe realizar la configuración de red necesaria para interconectar las máquinas virtuales (LAPTOP y VPS) y servicios entre sí.

## Note

En caso de disponer de un VPS en la nube, esta configuración de red no será necesaria, pues el hecho de que este en la nube, nos permitirá conectarnos a él desde cualquier máquina que disponga de conexión a Internet. Si es este tu caso, pasar directamente a la sección de Wireguard.

Se cuenta con dos máquinas virtuales: VPS y LAPTOP. Para montar este entorno, utilizaremos 3 interfaces de red que definiremos como adaptadores en cada una de las máquinas:

- **Red NAT:** será la encargada de dar internet a las máquinas (para la instalación de paquetes y descargas necesarias).
- **Red Interna:** será la que da visibilidad entre las máquinas LAPTOP y VPS (esta interfaz es necesaria porque se están utilizando máquinas virtuales, en un caso real, no sería necesario porque la visibilidad sería a través de Internet).
- **Red en modo Anfitrión:** esta permite interconectarse con la máquina host (se utilizará para acceder vía SSH a las máquinas).

Una vez se han creado las interfaces de red para las dos máquinas virtuales, se procede a explicar los pasos a seguir para la configuración de red en cada una de ellas.

## Configuración de Red en VPS

La configuración de red que se realizará en la máquina VPS, se hará a través de la herramienta y servicio [Netplan](#).

## Quote

Netplan es una herramienta que viene instalada en las versiones de Ubuntu a partir de la 17.10. Esta herramienta permite configurar al usuario una interfaz de red utilizando el formato **YAML**. Funciona junto a los *daemon* de red como **NetworkManager** y **systemd-networkd**, como interfaces para el kernel.

**Netplan** se encarga de leer la configuración indicada en los ficheros de la carpeta `/etc/netplan/*.yaml` y puede almacenar las configuraciones para todas las interfaces de red en estos archivos.

Al no disponer de entorno gráfico, esta será la opción más sencilla. Comenzamos creando o editando (en caso de existir) la plantilla por defecto de Netplan, en mi caso la encuentro en la ruta: `/etc/netplan/00-installer-config.yaml`

En este archivo YAML plasmaremos la siguiente configuración:

- 3 interfaces de red creadas en el paso anterior (los nombres son orientativos y no tienen por qué coincidir): **ens33 (NAT)**, **ens34 (Red Interna)** y **ens36 (Adaptador Puente)**.
- Deshabilitamos el DHCP en la interfaz ens34 (Red Interna), y definimos una dirección IPv4 estática.

```
network:  
  version: 2  
  ethernets:  
    ens33:  
      dhcp4: yes  
    ens34:  
      dhcp4: no  
      dhcp6: no  
      addresses: [10.10.10.1/24]  
    ens36:  
      dhcp4: yes
```

Tras esto, ejecutamos el siguiente comando para aplicar los cambios:

```
netplan apply
```

Y podemos comprobar que la configuración de red se ha aplicado correctamente con el comando: `ip a`

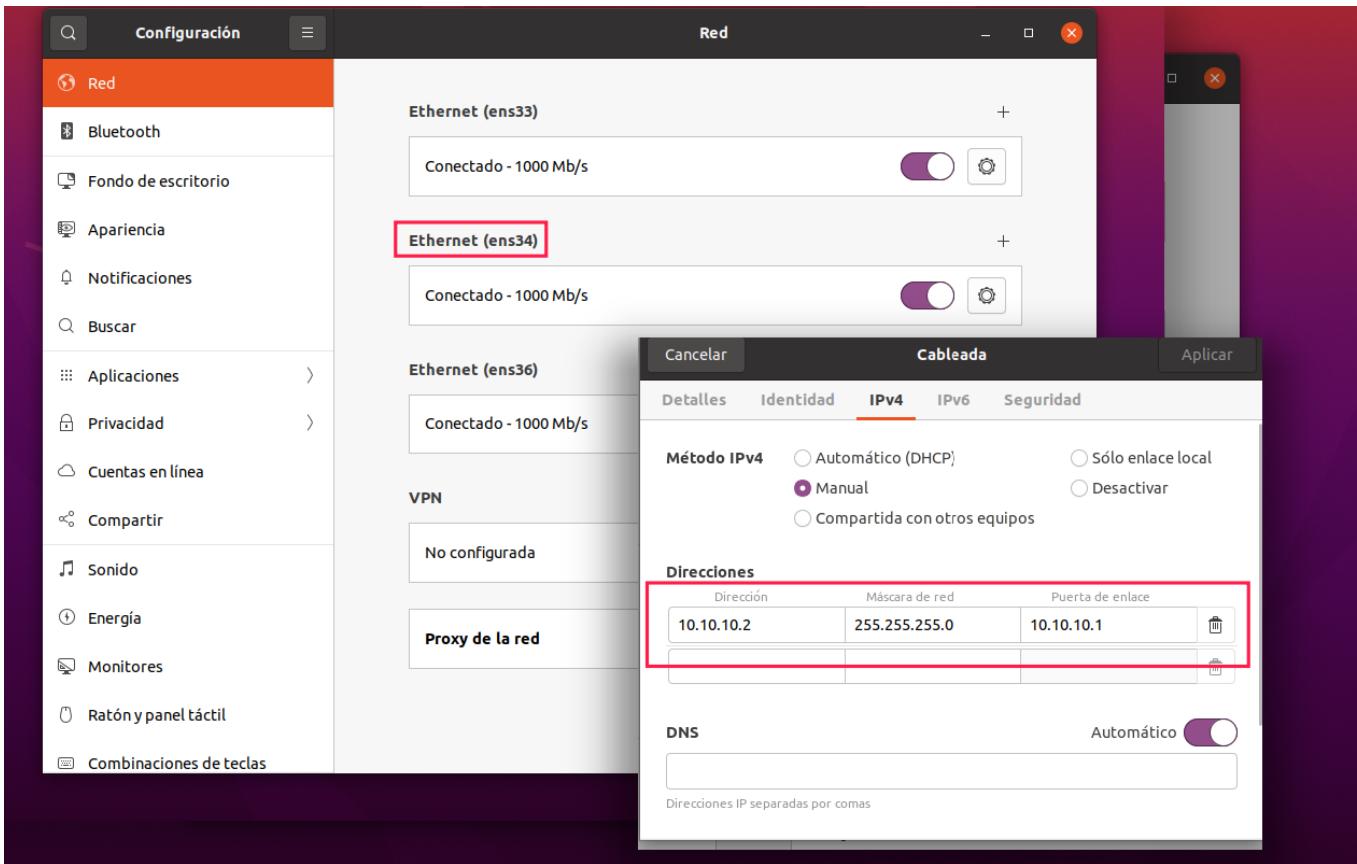
```
root@vps:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:61:a9:9a brd ff:ff:ff:ff:ff:ff
    inet 192.168.145.128/24 brd 192.168.145.255 scope global dynamic ens33
        valid_lft 1191sec preferred_lft 1191sec
    inet6 fe80::20c:29ff:fe61:a99a/64 scope link
        valid_lft forever preferred_lft forever
3: ens34: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:61:a9:a4 brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.1/24 brd 10.10.10.255 scope global ens34
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe61:a9a4/64 scope link
        valid_lft forever preferred_lft forever
4: ens36: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:61:a9:ae brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.134/24 brd 192.168.1.255 scope global dynamic ens36
        valid_lft 67047sec preferred_lft 67047sec
    inet6 fe80::20c:29ff:fe61:a9ae/64 scope link
        valid_lft forever preferred_lft forever
```

## Configuración de Red en LAPTOP

La configuración de red en la máquina LAPTOP, al contar con interfaz gráfica y dado que es un Ubuntu Desktop, será bastante sencilla.

Primero, vamos a la configuración de red, y encontraremos las 3 interfaces que hemos creado anteriormente. Nos situamos en la interfaz que corresponde a la Red Interna, en este caso **ens34**.

Y en la sección de IPv4, establecemos una IPv4 fija, junto con su máscara de red y la puerta de enlace.



Tras esto, simplemente verificamos con un: `ip a`, que las interfaces tienen asignadas las IPs que corresponden.

```
localadmin@laptop:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:c4:1f:a6 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.145.129/24 brd 192.168.145.255 scope global dynamic noprefixroute ens33
        valid_lft 1674sec preferred_lft 1674sec
    inet6 fe80::9655:f93:db8f:a678/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: ens34: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:c4:1f:b0 brd ff:ff:ff:ff:ff:ff
    altname enp2s2
    inet 10.10.10.2/24 brd 10.10.10.255 scope global noprefixroute ens34
        valid_lft forever preferred_lft forever
    inet6 fe80::70f2:eca0:6378:9d0c/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
4: ens36: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:c4:1f:ba brd ff:ff:ff:ff:ff:ff
    altname enp2s4
    inet 192.168.1.138/24 brd 192.168.1.255 scope global dynamic noprefixroute ens36
        valid_lft 62983sec preferred_lft 62983sec
    inet6 fe80::7ab0:c0b3:f8bd:8bb3/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

Llegados a este punto, se puede comenzar con el proceso de instalación de los paquetes y configuraciones necesarias para montar la Infraestructura.

# Servicio de VPN - Wireguard



## ¿Por qué Wireguard?

### jj Quote

WireGuard VPN es un software para **crear una red privada virtual (VPN)** extremadamente **sencilla de configurar, muy rápida** (más rápida que IPsec y OpenVPN) y que utiliza la **criptografía más moderna por defecto**, sin necesidad de seleccionar entre diferentes algoritmos de cifrado simétrico, asimétrico y de hashing. - [ref](#)

Wireguard presenta ciertas características que lo hacen preferible a otras soluciones de VPN conocidas como por ejemplo: OpenVPN. Los motivos por los cuales se ha decidido utilizar Wireguard para desplegar el servicio VPN de la organización son:

- La instalación y configuración de Wireguard es extremadamente sencilla. Y hay una gran cantidad de documentación y ejemplos tanto por parte de los desarrolladores como de la comunidad.
- Como se comentaba en su descripción, Wireguard es muy rápido, incluso más que otras posibles soluciones como por ejemplo Openvpn.
- Es multiplataforma, y dispone de aplicación para Windows, Linux y Darwin. También podemos encontrar aplicación móvil para Android y iOS.
- La gestión de clientes es muy sencilla y se pueden crear y eliminar nuevos perfiles con un par de clicks.

Y son estos puntos los que lo preceden a otras herramientas. A continuación se explicará el proceso de instalación y configuración de Wireguard (Servidor y Clientes).

## Instalación de Wireguard

La instalación de Wireguard es muy sencilla, ya que se puede encontrar el software en el gestor de paquetes de la mayoría de distribuciones Linux, o directamente en [la página oficial de Wireguard](#) en caso de utilizar otro sistema operativo o no encontrarse en los repositorios.

### Instalación en Linux

Para instalar el paquete de Wireguard, se realizarán las siguientes acciones desde una consola administrativa en las siguientes máquinas:

En la máquina que actua como servidor: **VPS**, se ejecuta:

```
apt update  
apt install -y wireguard
```

Y en la máquina que actua como cliente: **LAPTOP**, se ejecuta:

```
apt update  
apt install -y wireguard resolvconf
```

### Instalación en smartphone

Para realizar la instalación en un Smartphone, ya sea Android o iOS, se puede descargar la aplicación oficial de Wireguard tanto en Play Store como en la App Store:

## Configuración de Wireguard

Tras la instalación, el siguiente paso es configurar Wireguard. En primer lugar se configurará el Servidor de Wireguard en la máquina **VPS** y posteriormente los clientes Wireguard que se conecten al servidor: **LAPTOP**, etc.

Wireguard utiliza en el proceso de autenticación pares de claves tanto en el cliente como en el servidor, por tanto, el primer paso será realizar la creación de estas claves.

### Note

Durante el proceso de configuración de Wireguard, se trabajará sobre el directorio `/etc/wireguard` (que ha sido creado tras la instalación).

## Configuración de Servidor Wireguard

Primero, se realiza la configuración del Servidor de Wireguard. Sobre la máquina **VPS** se realiza la generación de claves: privada y pública del servidor.

Desde el directorio de Wireguard (`/etc/wireguard`), se ejecuta el comando:

```
wg genkey | tee private.key | wg pubkey > public.key
```

Tras la ejecución se debe obtener el par de claves generado (ambas en base64):

```
root@vps:/etc/wireguard# wg genkey | tee private.key | wg pubkey > public.key
root@vps:/etc/wireguard# cat private.key
KENQSQNZboacb5U34BwtMTqMLg1dffFNIjrUL71TF9mY=
root@vps:/etc/wireguard# cat public.key
6qazAiPNifKVY4EiLSkNSbKw4oM8xUclDTsNBX+JVk4=
```

### Warning

Es **MUY IMPORTANTE** no exponer el contenido de la clave privada del Servidor, pues cualquier persona podría generar nuevas claves como clientes y colarse en la red (en el caso se comparten pues es un entorno creado exclusivamente como POC).

A continuación se creará el archivo de configuración del servidor en `/etc/wireguard/wg0.conf`. El contenido de este archivo tiene el siguiente aspecto:

```
[Interface]
Address = <SERVER_IPV4_ADDRESS_KERE>
PrivateKey = <SERVER_PRIVATE_KEY_HERE>
ListenPort = <SERVER_LISTEN_PORT>
PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -A FORWARD -o %i -j ACCEPT; iptables -t nat -A POSTROUTING -o <LABNET_IFACE_HERE> -j MASQUERADE
```

```
PostDown = iptables -D FORWARD -i %i -j ACCEPT;iptables -D FORWARD -o %i -j ACCEPT; iptables -t nat -D POSTROUTING -o <LABNET_IFACE_HERE> -j MASQUERADE
```

En este archivo de configuración se puede encontrar:

- **Address:** indica la dirección IPv4 del servidor de Wireguard (dentro de la red VPN que se creará).
- **PrivateKey:** aquí figura el contenido de la clave privada del servidor (`/etc/wireguard/private.key`).
- **ListenPort:** número de puerto que se pondrá a la escucha para gestionar las conexiones de Wireguard entre cliente/s y servidor.
- **PostUp:** este es el comando que se ejecutará al levantar el servicio de Wireguard. Este comando hace la redirección que Wireguard necesita (en este campo figura el nombre de la interfaz de la red `LABNET`).
- **PostDown:** lo mismo que el PostUp pero al revés (en este campo figura el nombre de la interfaz de la red `LABNET`).

#### Note

Para más información acerca de los campos del archivo de configuración, consultar [las páginas man de wireguard](#)

En mi caso el archivo `/etc/wireguard/wg0.conf` quedaría de la siguiente forma:

```
[Interface]
Address = 192.168.10.1/24
PrivateKey = KENQSQNZboacb5U34BwtMTqMLg1dffNlJjrUL71TF9mY=
ListenPort = 41194
PostUp = iptables -A FORWARD -i %i -j ACCEPT;iptables -A FORWARD -o %i -j ACCEPT; iptables -t nat -A POSTROUTING -o ens34 -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT;iptables -D FORWARD -o %i -j ACCEPT; iptables -t nat -D POSTROUTING -o ens34 -j MASQUERADE
```

Después de esto, configuramos los permisos al directorio de Wireguard para limitar el acceso a los archivos del servidor exclusivamente al propio usuario root (para evitar que otros usuarios del servidor **VPS** puedan acceder a las claves). Ejecutamos el comando:

```
chmod 600 -R /etc/wireguard/
```

Finalmente habilitamos el servicio de Wireguard, para que se arranque al inicio, utilizando el siguiente comando:

```
systemctl enable wg-quick@wg0
```

### Note

Destacar que el nombre `wg0` se especifica en el servicio debido a que el archivo de configuración de Wireguard es `/etc/wireguard/wg0.conf`, en caso de utilizar otro nombre como archivo de configuración o tener múltiples, se puede indicar tras el `@` del comando anterior.

Y tras habilitar el servicio, lo iniciamos con:

```
systemctl start wg-quick@wg0
```

```
root@vps:/etc/wireguard# systemctl status wg-quick@wg0
● wg-quick@wg0.service - WireGuard via wg-quick(8) for wg0
   Loaded: loaded (/lib/systemd/system/wg-quick@.service; enabled; vendor preset: enabled)
     Active: active (exited) since Thu 2022-10-13 21:53:45 UTC; 1s ago
       Docs: man:wg-quick(8)
              man:wg(8)
              https://www.wireguard.com/
              https://www.wireguard.com/quickstart/
              https://git.zx2c4.com/wireguard-tools/about/src/man/wg-quick.8
              https://git.zx2c4.com/wireguard-tools/about/src/man/wg.8
     Process: 2384 ExecStart=/usr/bin/wg-quick up wg0 (code=exited, status=0/SUCCESS)
      Main PID: 2384 (code=exited, status=0/SUCCESS)

oct 13 21:53:45 vps systemd[1]: wg-quick@wg0.service: Succeeded.
oct 13 21:53:45 vps systemd[1]: Stopped WireGuard via wg-quick(8) for wg0.
oct 13 21:53:45 vps systemd[1]: Starting WireGuard via wg-quick(8) for wg0...
oct 13 21:53:45 vps wg-quick[2384]: [#/] ip link add wg0 type wireguard
oct 13 21:53:45 vps wg-quick[2384]: [#/] wg setconf wg0 /dev/fd/63
oct 13 21:53:45 vps wg-quick[2384]: [#/] ip -4 address add 192.168.10.1/24 dev wg0
oct 13 21:53:45 vps wg-quick[2384]: [#/] ip link set mtu 1420 up dev wg0
oct 13 21:53:45 vps wg-quick[2384]: [#/] iptables -A FORWARD -i wg0 -j ACCEPT; iptables -A FORWARD -o wg0 -j ACCEPT; iptables -t nat -A POSTROUTING -o ens34 -j ...
oct 13 21:53:45 vps systemd[1]: Finished WireGuard via wg-quick(8) for wg0.
lines 1-21/21 (END.)
```

## Configuración de Cliente Wireguard

Tras configurar el servidor de Wireguard en **VPS**, se procede a dar de alta los clientes de Wireguard o **peers**. El primer cliente que se dará de alta es en la máquina **LAPTOP** y será un ejemplo de un cliente Linux que accede a la VPN.

Lo primero será generar el par de claves igual que se hizo en el servidor de Wireguard, pero esta vez desde el propio cliente **LAPTOP**. Desde el directorio de Wireguard (`/etc/wireguard`), ejecutamos el comando:

```
wg genkey | tee private.key | wg pubkey > public.key
```

Después, se crea un archivo de configuración `/etc/wireguard/wg0.conf` (similar al que hemos creado en el servidor **VPS**). El contenido de este archivo presenta el siguiente formato:

```

[Interface]
Address = <CLIENT_IPV4_ADDRESS_HERE>
PrivateKey = <CLIENT_PRIVATE_KEY_HERE>
DNS = <SERVER_IPV4_ADDRESS_HERE>

[Peer]
PublicKey = <SERVER_PUBLIC_KEY_HERE>
Endpoint = <SERVER_PUBLIC_IP_HERE>:<SERVER_WG_PORT_HERE>
AllowedIPs = <WG_IPV4_RANGE_HERE>
PersistentKeepalive = 25

```

A continuación se explican los campos dentro de la interfaz y el peer:

- **Address:** la dirección IPv4 del cliente de Wireguard dentro de la red VPN (debe tener estar en el mismo rango de red definido en el servidor y no puede coincidir el **Address** puesto al servidor de Wireguard).
- **PrivateKey:** aquí figura el contenido de la clave privada del cliente que acabamos de generar (`/etc/wireguard/private.key`).
- **DNS:** en este campo se indica la dirección ip del servidor dns a utilizar. En nuestro caso vamos a poner la dirección IPv4 del servidor de Wireguard en la VPN debido a que utilizaremos más adelante nuestro propio dominio interno para la organización.
- **PublicKey:** aquí va el contenido de **la clave pública del SERVIDOR de Wireguard**.
- **Endpoint:** en este campo se indica la dirección donde esta el Servidor de Wireguard escuchando. Aquí pondremos la dirección ip "pública" del servidor **VPS** seguido del puerto donde escucha el Wireguard definido en la configuración.
- **AllowedIPs:** aquí se deben poner aquellos rangos de red a los cuales las peticiones deben ser enrutadas por Wireguard (por ejemplo, el rango de la VPN: `192.168.10.0/24`).
- **PersistentKeepalive:** este valor indica el número de segundos al que va a hacer ping al servidor de Wireguard para mantener la comunicación y conseguir así la bidireccionalidad del canal.

#### Note

Se especifica el campo **PersistentKeepalive** porque es posible que un Peer o cliente de Wireguard, esté detrás de un NAT o firewall que le impida recibir paquetes cuando no está enviando ninguno. Debido a que los NATs/firewalls permiten la comunicación bidireccional entre peer y servidor cuando es saliente desde el peer, pero en cambio, no es posible llegar desde el servidor hasta el peer. Debido a esto, el campo "PersistentKeepalive" mantiene la conexión activa en un intervalo de segundos razonable entre cliente y servidor permitiendo así recibir paquetes en el

cliente en cualquier momento. Para más información consultar [la documentación oficial de Wireguard](#)

En mi caso el archivo `/etc/wireguard/wg0.conf` del cliente de Wireguard **LAPTOP** quedaría de la siguiente forma:

```
[Interface]
Address = 192.168.10.2/32
PrivateKey = wE9QH3EZ1bNs4Q1Elh6sW2IEBLQLdcKS077uv2Qv82M=
DNS = 192.168.10.1

[Peer]
PublicKey = 6qazAiPNifKVY4EilSkNSbKw4oM8xUclDTsNBX+JVk4=
Endpoint = 10.10.10.1:41194
AllowedIPs = 192.168.10.0/24
PersistentKeepalive = 25
```

El siguiente paso es registrar las claves del nuevo peer en un nuevo directorio:

`/etc/wireguard/peers`, con el objetivo de tener un control/backup de las claves de los clientes y archivos de configuración.

#### Note

Este paso es **OPCIONAL**, pero viene muy bien en casos de migraciones de equipos y/o pérdidas de los archivos de configuración de los peers dados de alta. De esta forma, todo queda en el servidor de Wireguard. No es la opción más segura (de cara a los peers), pero si una opción que, a largo plazo, nos evitará quebraderos de cabeza.

En mi caso queda la siguiente jerarquía de archivos en el servidor de Wireguard:

```
root@vps:/etc/wireguard# tree /etc/wireguard/
/etc/wireguard/
├── peers
│   └── laptop
│       ├── private.key
│       ├── public.key
│       └── wg0.conf
└── wg0.conf

2 directories, 6 files
root@vps:/etc/wireguard#
```

Ahora toca dar de alta al nuevo peer en el Servidor de Wireguard. Esto se hace añadiendo la estructura que identifica al peer dentro del archivo de configuración de Wireguard, que se ve de esta forma:

```
# Client - <NAME>
[Peer]
PublicKey = <CLIENT_PUBLIC_KEY>
AllowedIPs = <CLIENT_WG_IPV4_ADDRESS>
```

Tras añadir el nuevo peer al archivo de configuración de Wireguard de **VPS**, se ve como:

```
[Interface]
Address = 192.168.10.1/24
PrivateKey = KENQSQNZboacb5U34BwtMTqMLg1dffNIjrUL71TF9mY=
ListenPort = 41194
PostUp = iptables -A FORWARD -i %i -j ACCEPT;iptables -A FORWARD -o %i -j
ACCEPT; iptables -t nat -A POSTROUTING -o ens34 -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT;iptables -D FORWARD -o %i -j
ACCEPT; iptables -t nat -D POSTROUTING -o ens34 -j MASQUERADE

# Client - LAPTOP
[Peer]
PublicKey = 9pFuhQ9s7WN9I3icj5JFRZwmKB3pf03qX+nMz2IfMRU=
AllowedIPs = 192.168.10.2/32
```

Finalmente, procedemos a reiniciar el servicio en cliente (**LAPTOP**) y servidor (**VPS**) para que se apliquen los cambios:

```
systemctl restart wg-quick@wg0
```

Tras esto se puede comprobar en el servidor de Wireguard que el cliente se ha registrado y conectado a la VPN exitosamente:

```
root@vps:/etc/wireguard# wg
interface: wg0
  public key: 6qazAiPNifKVY4EiLSkNSbKw4oM8xUc1DTsNBX+JVk4=
  private key: (hidden)
  listening port: 41194

peer: 9pFuhQ9s7WN9I3icj5JFRZwmKB3pf03qX+nMz2IfMRU=
  endpoint: 10.10.10.2:51556
  allowed ips: 192.168.10.2/32
  latest handshake: 3 seconds ago
  transfer: 180 B received, 92 B sent
root@vps:/etc/wireguard#
```

# Configuración de Cliente Wireguard en smartphone

Para dar de alta un cliente móvil en Wireguard, la forma más fácil y rápida es generar un QR del archivo de configuración del nuevo peer.

Para poder hacer esto, primero se debe instalar en **VPS** el paquete **qrencode**:

```
apt install qrencode
```

Tras esto, procedemos a generar un nuevo par de claves como se ha hecho anteriormente, sólo que esta vez lo haremos directamente desde el servidor de Wireguard:

```
mkdir -p /etc/wireguard/peers/phone
cd /etc/wireguard/peers/phone
wg genkey | tee private.key | wg pubkey > public.key
```

Tras generar las claves las trasladamos a un archivo de configuración del cliente en `/etc/wireguard/peers/phone/wg0.conf` que quedará de la siguiente forma:

```
[Interface]
Address = 192.168.10.3/32
PrivateKey = YH0HFwSCsHkGZ6Gkerh+fgueyf8Gd+GixjwPCi0bYXc=
DNS = 192.168.10.1

[Peer]
PublicKey = 6qazAiPNifKVY4EilSkNSbKw4oM8xUclDTsNBX+JVk4=
Endpoint = 10.10.10.1:41194
AllowedIPs = 192.168.10.0/24
PersistentKeepalive = 25
```

Damos de alta al nuevo peer en el servidor de Wireguard como hemos hecho anteriormente:

```
[Interface]
Address = 192.168.10.1/24
PrivateKey = KENQSQNzboacb5U34BwtMTqMLg1dfFNIjrUL71TF9mY=
ListenPort = 41194
PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -A FORWARD -o %i -j
ACCEPT; iptables -t nat -A POSTROUTING -o ens34 -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -D FORWARD -o %i -j
ACCEPT; iptables -t nat -D POSTROUTING -o ens34 -j MASQUERADE
```

```
# Client - LAPTOP
[Peer]
PublicKey = 9pFuhQ9s7WN9I3icj5JFRZwmKB3pf03qX+nMz2IfMRU=
AllowedIPs = 192.168.10.2/32

# Client - PHONE
[Peer]
PublicKey = Hz6fEvZhZxdzzRb05oRRvdasapDXva+pREdrU0atxc=
AllowedIPs = 192.168.10.3/32
```

Reiniciamos el servicio de Wireguard con:

```
systemctl restart wg-quick@wg0
```

Y verificamos que el nuevo peer se ha registrado correctamente:

```
root@vps:/etc/wireguard# wg
interface: wg0
  public key: 6qazAiPNifKVY4EiLSkNSbKw4oM8xUc1DTsNBX+JVk4=
  private key: (hidden)
  listening port: 41194

peer: 9pFuhQ9s7WN9I3icj5JFRZwmKB3pf03qX+nMz2IfMRU=
    allowed ips: 192.168.10.2/32

peer: Hz6fEvZhZxdzzRb05oRRvdasapDXva+pREdrU0atxc=
    allowed ips: 192.168.10.3/32
root@vps:/etc/wireguard# █
```

Finalmente, generamos un QR del archivo de configuración del nuevo peer con el comando:

```
qrencode -t ansiutf8 < /etc/wireguard/peers/phone/wg0.conf
```

```
root@vps:/etc/wireguard# qrencode -t ansiutf8 < /etc/wireguard/peers/phone/wg0.conf
```

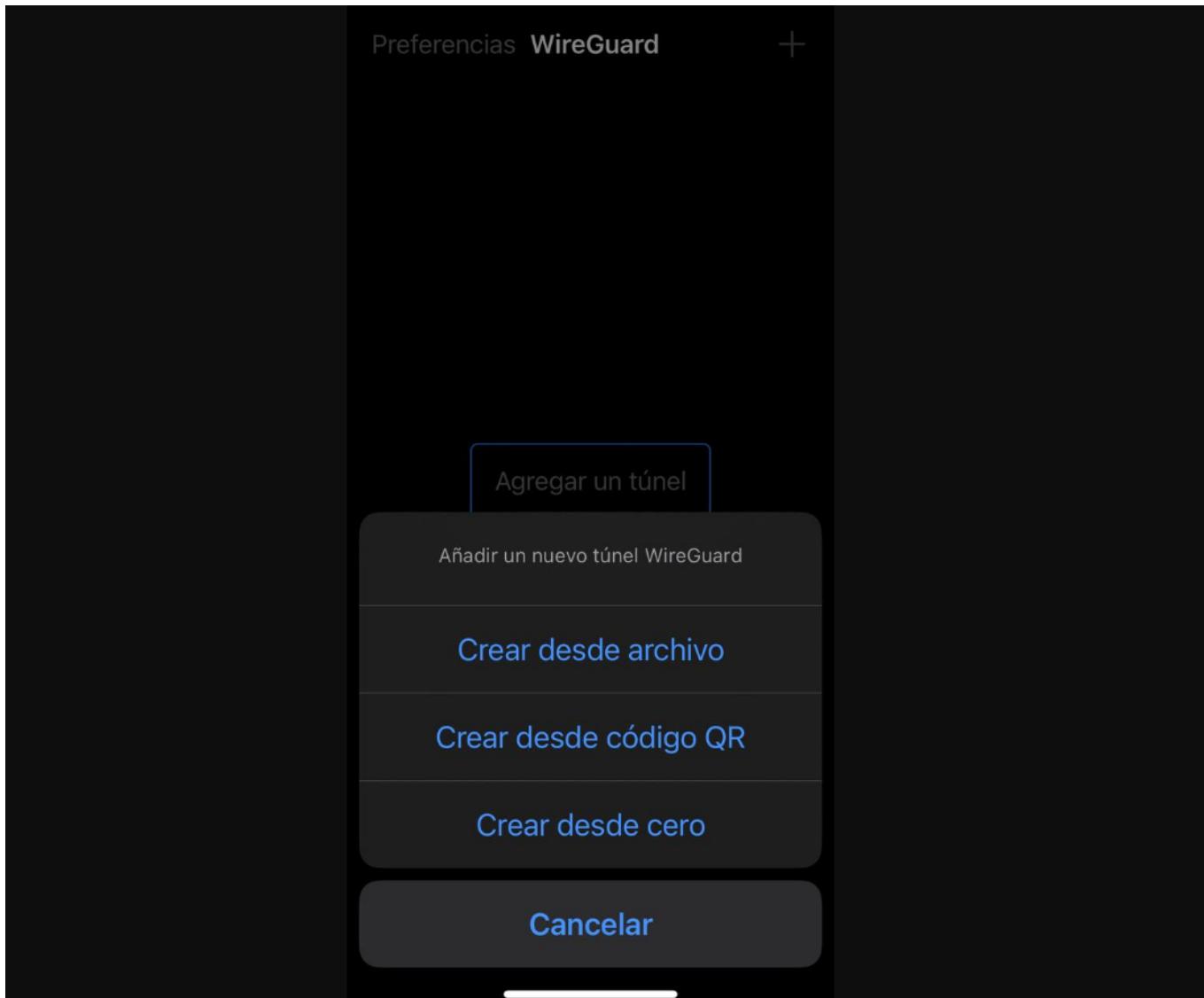


```
root@vps:/etc/wireguard#
```

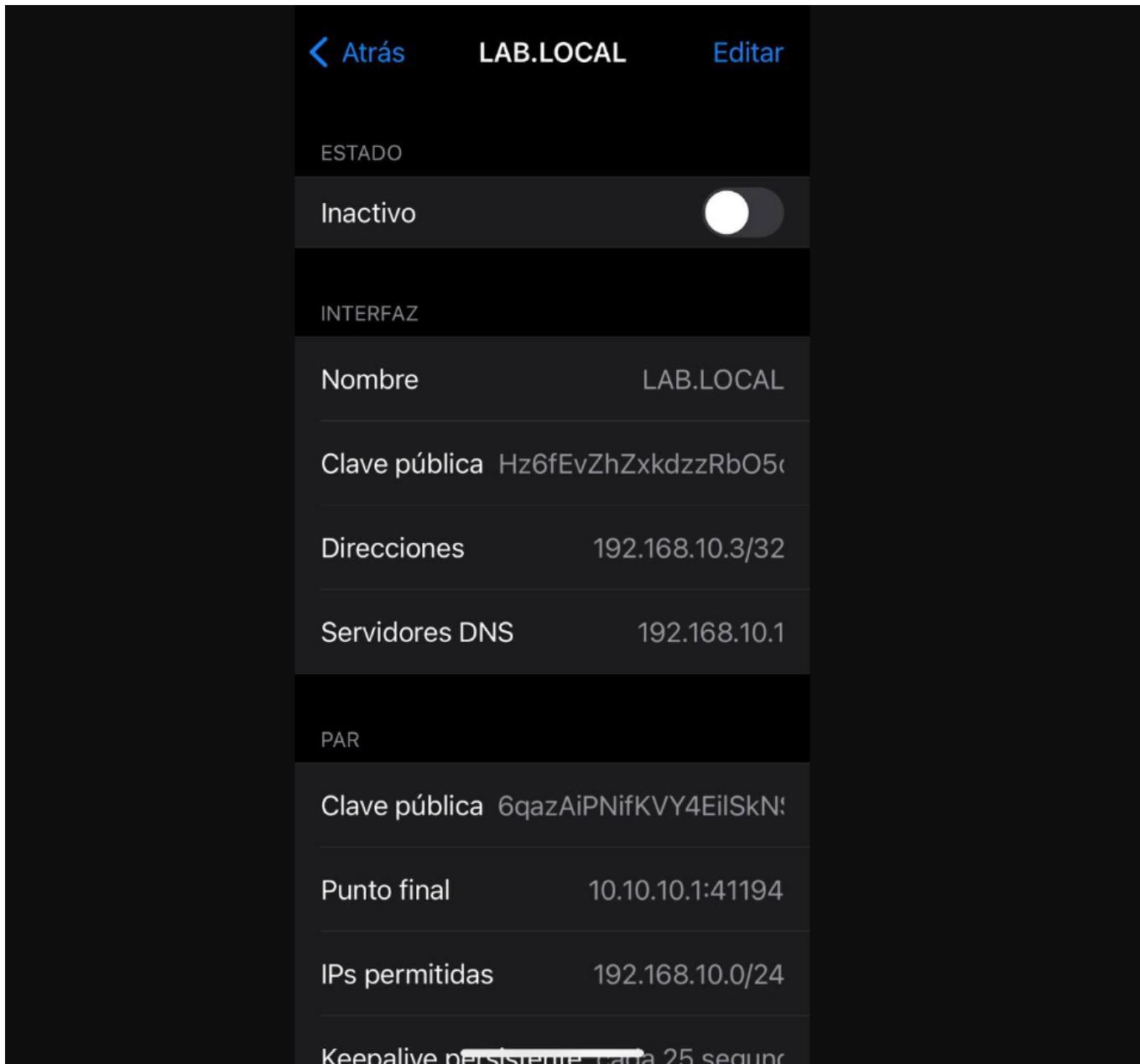
#### Note

En caso de haber problemas con el QR generado, también se puede exportar el qr a una imagen con el comando `qrencode -o output.png < /etc/wireguard/peers/phone/wg0.conf`, para poder descargarlo y visualizarlo.

Tras esto, desde la aplicación de Wireguard en el teléfono, se selecciona la opción de "Agregar un túnel", seguido de "Crear desde código QR":



Y al escanear el QR, se pedirá un nombre para el tunel. Después podrá encontrarse la configuración en el listado de la aplicación lista para usarse:



## Pruebas de Conexión

Cómo último paso de esta sección, comprobaremos que la conexión entre cliente y servidor de Wireguard (**LAPTOP** y **VPS**) funciona correctamente.

```
root@vps:/etc/wireguard# wg
interface: wg0
  public key: 6qazAiPNifKVY4EilSkNSbKw4oM8xUclDTsNBX+JVk4=
  private key: (hidden)
  listening port: 41194

peer: 9pFuQ9s7WN9I3icj5JFRZwmKB3pf03qX+nMz2IfMRU=
  endpoint: 10.10.10.2:54604
  allowed ips: 192.168.10.2/32
  latest handshake: 20 seconds ago
  transfer: 4.43 KiB received, 4.57 KiB sent

peer: Hz6fEvZhZxdzzRb05oRRvdasapDXva+pREdrU0atxc=
  allowed ips: 192.168.10.3/32
root@vps:/etc/wireguard# python3 -m http.server --bind 192.168.10.1
Serving HTTP on 192.168.10.1 port 8000 (http://192.168.10.1:8000/) ...
192.168.10.2 - - [15/Oct/2022 11:48:17] "HEAD / HTTP/1.1" 200 -
[

root@laptop:/etc/wireguard# curl -I http://192.168.10.1:8000/
HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/3.8.10
Date: Sat, 15 Oct 2022 11:48:17 GMT
Content-type: text/html; charset=utf-8
Content-Length: 467

root@laptop:/etc/wireguard#
```

Y, como podemos ver, la conexión a través de la VPN funciona correctamente.

El siguiente paso consiste en instalar y configurar el servidor de DNS Interno que utilizaremos dentro de la infraestructura para acceder a los distintos servicios de la organización.

# Servicio de DNS Interno - DNSmasq



## ¿Por qué utilizar DNSmasq y no otros servidores DNS?

jj Quote

[DNSmasq](#) proporciona un servidor DNS, un servidor DHCP con soporte para DHCPv6 y PXE, y un servidor TFTP. Está diseñado para ser liviano y consumir poco, adecuado para enrutadores y cortafuegos con recursos limitados. dnsmasq también se puede configurar para almacenar en caché las consultas a DNS con el fin mejorar las velocidades de búsqueda de DNS de los sitios visitados con anterioridad. [ref](#)

Dnsmasq es el servidor de DNS ideal para utilizar en este caso. Pues, realmente, hay mejores opciones (o más completas), como por ejemplo Bind9; no obstante, la utilidad que le vamos a dar al servidor DNS es tan simple que no merece la plena complicarse con otros programas más complejos.

Se utilizará DNSmasq simplemente como servicio de resolución de nombres dentro de la organización. A través de nuestro servidor VPN, fijaremos la dirección donde escucha DNSmasq y este será el encargado de resolver todos los nombres de dominio que consulten los clientes.

Y este motivo es, en si, la justificación para utilizar DNSmasq como servidor DNS de la organización. A continuación se procederá a explicar su instalación, configuración e integración con otros servicios.

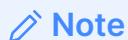
## Instalación de DNSmasq

La instalación de DNSmasq es muy sencilla, ya que el paquete se encuentra en los repositorios de cualquier distribución.

Antes de comenzar con la instalación de DNSmasq sobre el servidor VPN (**VPS**), debemos deshabilitar el servicio `systemd-resolve`, pues este servicio es el que se encarga de hacer la resolución de nombres en la máquina y viene por defecto en Ubuntu >18.04.

Para deshabilitar el servicio y liberar el puerto 53, se ejecutan los siguientes comandos:

```
systemctl disable systemd-resolved  
systemctl stop systemd-resolved  
unlink /etc/resolv.conf  
echo "nameserver 8.8.8.8" | tee /etc/resolv.conf
```



Es necesario eliminar el enlace simbólico de `/etc/resolv.conf` ya que apunta a un archivo de configuración de `systemd-resolve`. En su lugar se creará uno nuevo con los dns de Google (posteriormente se editará).

Tras esto, ya se puede instalar dnsmasq en el servidor con:

```
apt install -y dnsmasq
```

Tras la instalación se deberá encontrar el archivo `/etc/dnsMasq.conf` que utilizaremos para realizar la configuración en el siguiente apartado.

## Configuración de DNSmasq

La configuración de DNSmasq es muy sencilla, ya que todo se limita a un archivo de configuración situado en: `/etc/dnsMasq.conf`

Este archivo, a pesar de ser muy extenso, sólamente tendremos que descomentar las siguientes líneas y dejarlas como a continuación se detalla:

```
# Listen on this specific port instead of the standard DNS port  
# (53). Setting this to zero completely disables DNS function,  
# leaving only DHCP and/or TFTP.  
port=53
```

```
# Never forward plain names (without a dot or domain part)
domain-needed

# Never forward addresses in the non-routed address spaces.
bogus-priv

# By default, dnsmasq will send queries to any of the upstream
# servers it knows about and tries to favour servers to are known
# to be up. Uncommenting this forces dnsmasq to try each query
# with each server strictly in the order they appear in
# /etc/resolv.conf

strict-order

# Set this (and domain: see below) if you want to have a domain
# automatically added to simple names in a hosts-file.

expand-hosts

# Set the domain for dnsmasq. this is optional, but if it is set, it
# does the following things.
# 1) Allows DHCP hosts to have fully qualified domain names, as long
#    as the domain part matches this setting.
# 2) Sets the "domain" DHCP option thereby potentially setting the
#    domain of all systems configured by DHCP
# 3) Provides the domain part for "expand-hosts"

domain=lab.local

# Or which to listen on by address (remember to include 127.0.0.1 if
# you use this.)
listen-address=192.168.10.1,127.0.0.1
```

Con esta configuración, pondremos a escuchar el servidor DNS en el puerto 53 local y la dirección de la VPN, gestionando el dominio de "lab.local" y definiendo los dns en el archivo `/etc/hosts`.

Después reiniciamos el servicio de DNSmasq con el comando:

```
systemctl restart dnsmasq
```

Tras esto, editamos el archivo `/etc/hosts` del servidor **VPS** introduciendo las siguientes líneas para añadir los nombres de las máquinas de la VPN y un subdominio interno a modo de prueba:

```
# LAB.LOCAL DNS records
192.168.10.1 vps.lab.local
192.168.10.2 laptop.lab.local

192.168.10.1 test.lab.local
```

Tras esto, se edita de nuevo el archivo `/etc/resolv.conf` para que resuelva primero el archivo hosts local y, como segundo nameserver, el de Google:

```
nameserver 127.0.0.1
nameserver 8.8.8.8
nameserver 8.8.4.4
```

Y volvemos a reiniciar el servicio de DNSmasq:

```
systemctl restart dnsmasq
```

Para comprobar que se han aplicado los cambios se realiza una consulta dns localmente:

```
root@vps:/etc/wireguard# dig a test.lab.local

; <>> DiG 9.16.1-Ubuntu <>> a test.lab.local
;; global options: +cmd
;; Got answer:
;; WARNING: .local is reserved for Multicast DNS
;; You are currently testing what happens when an mDNS query is leaked to DNS
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59150
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;test.lab.local.           IN      A

;; ANSWER SECTION:
test.lab.local.        0       IN      A      192.168.10.1

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: sáb oct 15 12:13:59 UTC 2022
;; MSG SIZE  rcvd: 59

root@vps:/etc/wireguard# █
```

## Configuración de DNS a través de Wireguard

Para que los peers/clientes de Wireguard puedan resolver dominios internos utilizando el servidor DNS de **VPS**, se debe editar la configuración del archivo de configuración de Wireguard.

Como nosotros ya lo hicimos a la hora de crear el nuevo Peer, simplemente se destaca la linea de la configuración que hace efecto:

```
DNS: 192.168.10.1
```

## Pruebas de resolución

Tras haber concluido la configuración en servidor y clientes, se procede a realizar una prueba de resolución del dominio de prueba creado para este propósito. Para este caso se realiza una resolución DNS desde el cliente **LAPTOP** con el siguiente comando:

```
dig A test.lab.local
```

```
root@laptop:/etc/wireguard# dig a test.lab.local

; <>> DiG 9.16.1-Ubuntu <>> a test.lab.local
;; global options: +cmd
;; Got answer:
;; WARNING: .local is reserved for Multicast DNS
;; You are currently testing what happens when an mDNS query is leaked to DNS
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 9212
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;test.lab.local.          IN      A

;; ANSWER SECTION:
test.lab.local.      0      IN      A      192.168.10.1

;; Query time: 0 msec
;; SERVER: 192.168.10.1#53(192.168.10.1)
;; WHEN: sáb oct 15 14:15:23 CEST 2022
;; MSG SIZE  rcvd: 59

root@laptop:/etc/wireguard#
```

Con esto se comprueba que las resoluciones DNS del dominio interno **LAB.LOCAL** se están realizando a través de la VPN y los nombres de dominio se están resolviendo correctamente.

Con el servidor DNS activo y funcionando, se procederá a instalar el servidor web que actuará como proxy inverso entre el servidor y los contenedores de aplicaciones.

# Servicio de Proxy Inverso - Nginx



## ¿Por qué utilizar Nginx?

» Quote

[NGINX](#), pronunciado en inglés como «engine-ex», es un famoso software de servidor web de código abierto. En su versión inicial, funcionaba en servidores web HTTP. Sin embargo, hoy en día también sirve como proxy inverso, balanceador de carga HTTP y proxy de [correo](#) electrónico para IMAP, POP3 y SMTP. [ref](#)

Nginx, además de ser un servidor web muy utilizado para gestión de cualquier aplicación o servicio web, también es un candidato ideal para gestionar el tráfico web de la infraestructura.

Existen muchas alternativas a Nginx, como por ejemplo: Apache, Caddy, Lighttpd, Litespeed, etc. Para este entorno, realmente se podría utilizar cualquiera de las alternativas, en especial Caddy, pero los motivos por el cual he decidido utilizar Nginx es porque estoy bastante familiarizado con la sintaxis y me parece muy limpia la forma que tiene de organizar los archivos de configuración, así como que hay muchos ejemplos para desplegar cualquier configuración.

No hay más, además de las cuestiones obvias de que es ligero, fácil de instalar, fácil de configurar y ampliamente conocido.

A continuación se procede con el proceso de instalación y configuración de Nginx.

## Instalación de Nginx

De igual forma que con Wireguard y Dnsmasq, la instalación de Nginx es trivial, ya que también se encuentra en los repositorios.

```
apt install -y nginx
```

Tras la instalación nos moveremos al directorio `/etc/nginx` dónde procederemos con la configuración.

## Configuración de Nginx

El primer paso consiste en crear un certificado autofirmado, que será el que utilicemos para nuestros subdominios internos. Para crearlo se ejecuta el siguiente comando:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout  
/etc/ssl/private/nginx-selfsigned.key -out /etc/ssl/certs/nginx-  
selfsigned.crt
```

Después se genera un Diffie-Hellman robusto que se utilizará para la negociación con los clientes (la generación puede demorar unos minutos):

```
openssl dhparam -out /etc/nginx/dhparam.pem 4096
```

Se continua creando el archivo `/etc/nginx/snippets/self-signed.conf` con el siguiente contenido:

```
ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;  
ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;
```

Tras esto, se crea el archivo `/etc/nginx/snippets/ssl-params.conf` con el siguiente contenido:

```
ssl_protocols TLSv1.2;  
ssl_prefer_server_ciphers on;  
ssl_dhparam /etc/nginx/dhparam.pem;  
ssl_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-RSA-  
AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384;  
ssl_ecdh_curve secp384r1; # Requires nginx >= 1.1.0  
ssl_session_timeout 10m;  
ssl_session_cache shared:SSL:10m;  
ssl_session_tickets off; # Requires nginx >= 1.5.9  
ssl_stapling on; # Requires nginx >= 1.3.7  
ssl_stapling_verify on; # Requires nginx >= 1.3.7  
resolver 8.8.8.8 8.8.4.4 valid=300s;  
resolver_timeout 5s;  
# Disable strict transport security for now. You can uncomment the following  
# line if you understand the implications.  
# add_header Strict-Transport-Security "max-age=63072000; includeSubDomains;  
preload";
```

```
add_header X-Frame-Options DENY;
add_header X-Content-Type-Options nosniff;
add_header X-XSS-Protection "1; mode=block";
```

Después de haber definido los archivos de configuración del SSL para Nginx, se hace una copia del archivo de configuración por defecto de nginx situado en `/etc/nginx/sites-available/default` por el siguiente:

```
server {
    listen 192.168.10.1:80;

    root /var/www/html;

    index index.html index.htm index.nginx-debian.html;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

#### Note

Se pone el nginx a escuchar en la dirección de la VPN (192.168.10.1) debido a que, el acceso a los servicios web internos, debe ser accesible únicamente vía VPN y así evitamos exponer el puerto de Nginx a Internet.

Continuando con la creación de subdominios internos, como prueba de concepto se creará un virtual host para el subdominio "test.lab.local" (que ya habíamos definido previamente):

```
cp /etc/nginx/sites-available/default /etc/nginx/sites-
available/test.lab.local
```

El contenido de este archivo debe quedar como el siguiente:

```
server {
    listen 192.168.10.1:80;
    server_name test.lab.local;
    # Redirige a la URL https://test.lab.local
    return 301 https://$host$request_uri;
```

```
# Redirect all non-https requests
rewrite ^ https://$host$request_uri? permanent;
}

server {
    listen 192.168.10.1:443 ssl http2;

    include snippets/self-signed.conf;
    include snippets/ssl-params.conf;

    server_name test.lab.local;

    client_max_body_size 0;
    proxy_http_version 1.1;
    proxy_request_buffering off;

    location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Como puede apreciarse, se define un bloque para redirigir todas las peticiones HTTP (80) a HTTPS (443), en el segundo bloque se incluyen todos los archivos de configuración SSL generados previamente y se realiza un **proxy\_pass** contra la dirección local: `http://127.0.0.1:8000`

#### Note

Si el puerto es https en lugar de http, se puede hacer el proxy\_pass como https.

De esta forma se utiliza Nginx como un reverse proxy para gestionar los diferentes servicios web internos así como los subdominios que apuntan a dichos servicios web.

Para activar el virtual host creado, se crea un enlace simbólico al directorio de **sites-enabled** con el comando:

```
ln -s /etc/nginx/sites-available/test.lab.local /etc/nginx/sites-enabled
```

Finalmente, se comprueba que la sintaxis del archivo de configuración creado es correcta con:

```
nginx -t
```

### Note

Aunque salen unos warnings son debido a que el certificado utilizado se trata de un certificado autofirmado, así que, procederemos a ignorarlo

Tras esto, se habilita y reinicia el servicio de nginx con:

```
systemctl enable nginx
systemctl restart nginx
```

## Pruebas de conexión

Para comprobar que la configuración de Nginx ha sido correcta, se procede a levantar un servidor web en la dirección local <http://127.0.0.1:8000>.

Tras esto, se comprueba desde la máquina **LAPTOP** que, al realizar una petición HTTP al subdominio interno **test.lab.local** sucede que:

1. Wireguard enruta la petición DNS por la VPN.
2. Dnsmasq resuelve correctamente el subdominio interno devolviendo la dirección IP de la máquina.
3. Nginx enruta la petición hacia la dirección local.
4. Se obtiene la respuesta HTTP esperada en el lado del cliente.

```
root@vps:/etc/nginx# python3 -m http.server --bind 127.0.0.1
Serving HTTP on 127.0.0.1 port 8000 (http://127.0.0.1:8000/) ...
127.0.0.1 - - [15/Oct/2022 13:29:50] "HEAD / HTTP/1.1" 200 -
```

```
root@laptop:/home/localadmin# curl -I -k https://test.lab.local
HTTP/2 200
server: nginx/1.18.0 (Ubuntu)
date: Sat, 15 Oct 2022 13:29:50 GMT
content-type: text/html; charset=utf-8
content-length: 1108
x-frame-options: DENY
x-content-type-options: nosniff
x-xss-protection: 1; mode=block
root@laptop:/home/localadmin#
```

Tras verificar que la prueba es correcta, se elimina el subdominio del archivo `/etc/hosts` y también el virtual host correspondiente en `/etc/nginx/sites-available/test.lab.local` así como su enlace simbólico en `/etc/nginx/sites-enabled/test.lab.local`.

El siguiente paso será comenzar con el despliegue de los contenedores de aplicaciones. En la siguiente sección se mostrará el diseño escogido para la estructuración de contenedores, así como la guía de despliegue de cada uno de ellos.

## Servicio de Orquestación de Aplicaciones - Docker



### ¿Por qué usar Docker para el despliegue de aplicaciones?

Un tema que siempre se plantea antes de comenzar el diseño y montaje de una Infraestructura, es ¿cómo se va a llevar a cabo el proceso de despliegue de todos los servicios sin que esto cause una gran problemática? y más cuando son diferentes tecnologías, con diferentes versiones o dependencias particulares.

Ante este problema, aparece una solución efectiva y directa, y es utilizar el **despliegue contenerizado** a través de la herramienta [Docker](#)

Docker te permite crear un **contenedor** (que es algo así como una especie de "chroot jail") en el que reutilizas el kernel del host pero en dicho entorno se incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, etc).

Esta funcionalidad te permite desplegar distintas aplicaciones aisladas entre si, de forma programática y, por tanto, automática, aportando así una gran confianza de que tu aplicación va a funcionar con certeza.

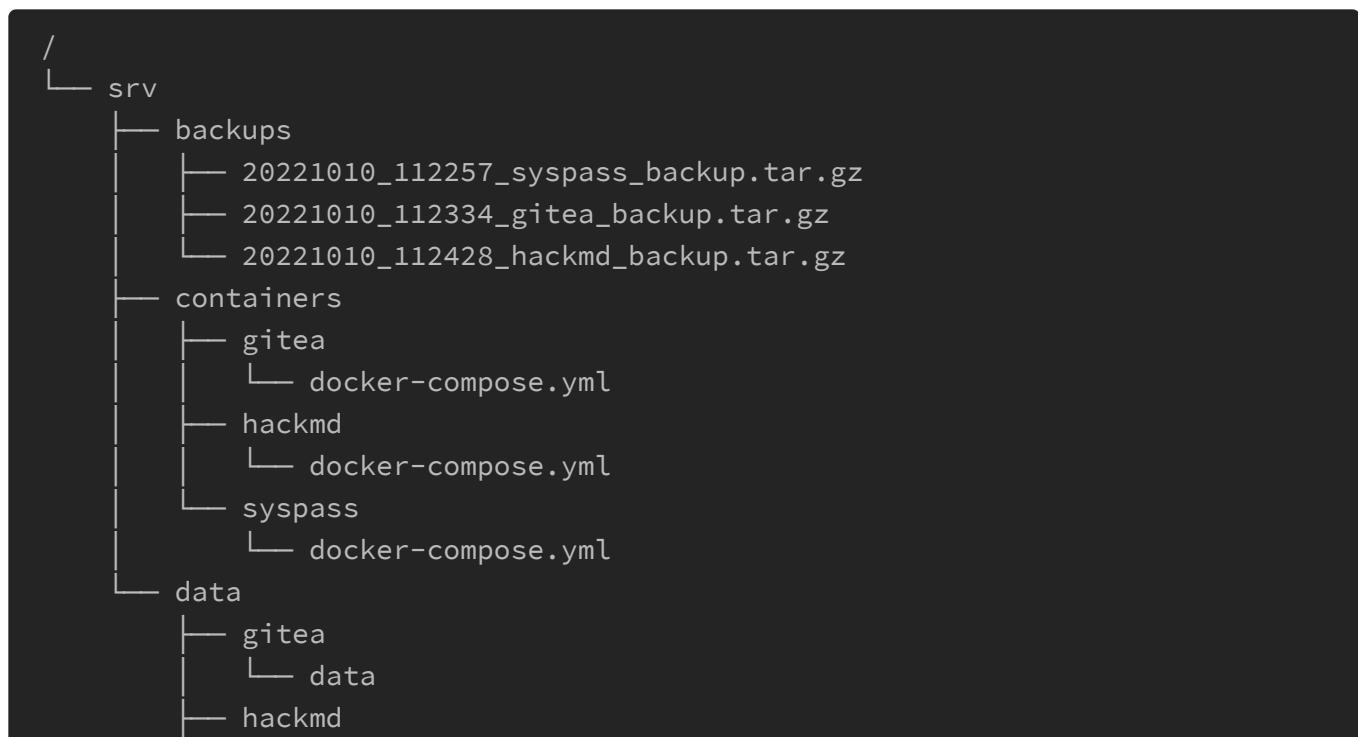
Es aislamiento también permite que, si se compromete una aplicación dentro de un contenedor, no se pueda saltar al resto de contenedores (a priori).

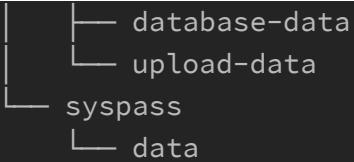
En sí, no hay motivo para no utilizarlo, ya que no existe una alternativa (o al menos no se ha encontrado), que permita realizar este tipo de despliegues. Por tanto, en los siguientes puntos se explicará tanto el diseño que se ha seguido para estructurar los archivos de cada aplicación, así como la forma de proceder al despliegue de las mismas.

## Jerarquía de aplicaciones dockerizadas

En primer lugar, es necesario definir una jerarquía de directorios y archivos que nos permitirá orquestar la creación y despliegue de los contenedores así como el proceso de backups que asegure la información contenida.

Un esquema de la jerarquía del sistema de archivos propuesta, sería la siguiente:





A continuación se explica brevemente cada uno de los 3 directorios principales:

- **Containers:** en este directorio se situarán los archivos necesarios para poder desplegar el/los contenedor/es de cada uno de los servicios. Para hacerlo muy sencillo y poder automatizar el proceso de backups, se utilizarán archivos `docker-compose.yml` en el directorio de cada servicio. en este directorio se localizan los archivos necesarios para el despliegue de cada aplicación dockerizada (Dockerfiles, Docker-Composes, etc). Con el objetivo de poder automatizar el despliegue y los backups de los contenedores, se creará un archivo `docker-compose.yml` en el directorio raíz
- **Data:** en este directorio se situarán los datos de los volúmenes de cada servicio desplegado. Es decir, cuando se despliega un contenedor, si existe información a guardar como por ejemplo una base de datos, archivos de usuarios o similar, se define en el `docker-compose.yml` un volumen que permitirá que estos datos persistan a los reinicios de los contenedores en un directorio de la máquina host.
- **Backups:** este directorio se utilizará en el proceso de backups donde se generarán los archivos de las copias realizadas a los distintos contenedores de las aplicaciones.

Por otro lado, al desplegar cada uno de los servicios en contenedores docker, será necesario bindearlos a un puerto para poder llegar a ellos desde el Nginx (actuando como proxy inverso). Para ello se ha definido también la siguiente distribución de puertos y servicios que se plasma en la siguiente tabla:

| Aplicación | Dirección IP | Puerto | Comentario       |
|------------|--------------|--------|------------------|
| Gitea      | 127.0.0.1    | 8000   | Servicio Web     |
| Gitea      | 192.168.10.1 | 2222   | Servicio SSH     |
| Hackmd     | 127.0.0.1    | 8001   | Servicio Web     |
| Hackmd     | 127.0.0.1    | 5432   | Postgres DB      |
| Zulip      | 127.0.0.1    | 8002   | Servicio Web     |
| Zulip      | 127.0.0.1    | 8443   | Servicio Web SSL |
| Syspass    | 127.0.0.1    | 8444   | Servicio Web SSL |
| Wekan      | 127.0.0.1    | 8003   | Servicio Web     |
| WikiJS     | 127.0.0.1    | 8004   | Servicio Web     |

| Aplicación | Dirección IP | Puerto | Comentario       |
|------------|--------------|--------|------------------|
| Jitsi Meet | 127.0.0.1    | 8445   | Servicio Web SSL |
| Jitsi Meet | 127.0.0.1    | 8080   | Servicio JVB     |
| Jitsi Meet | 127.0.0.1    | 10000  | Servicio JVB     |

## Instalación de Docker y Docker-Compose

La instalación de **Docker** y **Docker-Compose** a diferencia de los paquetes anteriores, varía un poco en cuestión a dificultad, pero su instalación es bastante guiada y sólo hay que seguir los pasos de la [documentación oficial](#)

La instalación de Docker y Docker Compose se realizará en el servidor **VPS**, que será donde se desplieguen cada una de las aplicaciones de la infraestructura. Los pasos de instalación se realizarán disponiendo de una consola con privilegios de superusuario o root.

Primero, se actualizan los repositorios y se instalan los paquetes necesarios para poder utilizar `apt` sobre HTTPS:

```
apt update
apt install -y ca-certificates curl gnupg lsb-release
```

Se añade la clave PGP oficial de Docker con:

```
mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
```

Se añade el repositorio de Docker al directorio de sources.list:

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list >
/dev/null
```

Con esto ya hemos preparado los repositorios de Docker, ahora se procede a su instalación:

```
apt update
apt install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

Tras la ejecución del comando, se habrá instalado en la máquina las últimas versiones de: Docker Engine, containerd y Docker Compose.

Para verificar que la instalación ha sido correcta, se verifica desplegando el contenedor de "Hello World" de Docker con:

```
systemctl start docker  
docker run hello-world
```

```
root@vps:/home/localadmin# docker run hello-world  
  
Hello from Docker!  
This message shows that your installation appears to be working correctly.  
  
To generate this message, Docker took the following steps:  
1. The Docker client contacted the Docker daemon.  
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
(amd64)  
3. The Docker daemon created a new container from that image which runs the  
executable that produces the output you are currently reading.  
4. The Docker daemon streamed that output to the Docker client, which sent it  
to your terminal.  
  
To try something more ambitious, you can run an Ubuntu container with:  
$ docker run -it ubuntu bash  
  
Share images, automate workflows, and more with a free Docker ID:  
https://hub.docker.com/  
  
For more examples and ideas, visit:  
https://docs.docker.com/get-started/  
  
root@vps:/home/localadmin# docker -v  
Docker version 20.10.19, build d85ef84  
root@vps:/home/localadmin# docker compose version  
Docker Compose version v2.11.2  
root@vps:/home/localadmin#
```

## Despliegue de aplicaciones contenerizadas

En esta sección se explicarán los pasos a seguir para desplegar algunas aplicaciones que pueden resultar de utilidad en una infraestructura empresarial o en una organización colaboración privada.

Como bien se ha comentado anteriormente, al disponer de diferentes tecnologías con diferentes dependencias y requerimientos, la mejor opción es apostar por un despliegue: rápido, seguro y automático. Esto es posible gracias a Docker y la capacidad de aislar las aplicaciones entre si, creando entornos únicos e independientes donde corren cada una de las aplicaciones y no interfieren con el sistema operativo base.

# Servicio de Control de Versiones - Gitea



# Gitea

## ¿Por qué Gitea?

Gitea es una aplicación web para el control de versiones en proyectos de desarrollo. Esta desarrollada en [Golang](#) y utiliza [Hugo](#) como generador y renderizador de plantillas estáticas.

### » Quote

El objetivo de Gitea es: proporcionar la forma más fácil, rápida e indolora de configurar un servicio Git on-premises. - cita de los propios desarrolladores en su [página oficial](#)

Gitea es similar a otros servicios conocidos de control de versiones como Github, Gitlab o Bitbucket. Y existe la posibilidad de [probar el servicio online](#)

Frente a estas soluciones tan conocidas y extendidas surge la pregunta... ¿por qué utilizar Gitea? por ejemplo... frente a Gitlab (conocido servicio de control de versiones que también permite el despliegue on-premises).

Tras haber probado tanto Gitlab como Gitea durante un tiempo, he llegado a las siguientes conclusiones que, me priorizan Gitea sobre Gitlab:

1. Gitea utiliza **Hugo** para servir las vistas del servicio web, una característica de Hugo, es que es insultantemente rápido, pues genera contenido estático que es cargado con mayor velocidad en comparación a Gitlab.

2. Gitea utiliza como base de datos por defecto **Sqlite3** (aunque permite utilizar otros motores de bases de datos como Mysql, Postgres o Mssql), el uso de Sqlite3 es idóneo para una infraestructura pequeña-media pues la carga y escritura de información se realiza de forma muy rápida y resulta bastante sencilla de manejar.
3. Gitea **requiere de muy poca memoria ram y cpu** para su uso, a diferencia de Gitlab cuyos requerimientos mínimos son bastante superiores (4GB minimo para gestionar 500 usuarios). Obviamente, Gitlab consume más recursos debido a que también proporciona mayor cantidad de funcionalidades (muchas de las que Gitea no ofrece).
4. Gitea, a diferencia de Gitlab, **es mucho más simple**, esto es una ventaja por un lado y también desventaja, pues si bien la sencillez aporta mayor velocidad y menor complejidad, también se ve ausente de algunas funcionalidades interesantes que estan presentes en Gitlab.
5. Finalmente, Gitea **cubre la mayoría de las funcionalidades esperadas en un servicio de control de versiones**. Es decir, es ideal para comenzar con desarrollos sencillos y a colaborar entre los distintos miembros de la organización.

Debido a estas razones y a su facilidad para desplegar el contenedor, se ha elegido Gitea como servicio de control de versiones para la infraestructura.

## Despliegue de contenedor de Gitea

Para desplegar el contendor de Gitea de forma automatizada se utilizará el siguiente archivo YAML de Docker-Compose:

```
version: "3"
services:
  gitea:
    image: gitea/gitea:1.17.3
    container_name: gitea
    environment:
      - USER_UID=1000
      - USER_GID=1000
    restart: always
    networks:
      - gitea
    volumes:
      - /srv/data/gitea/data:/data
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro
    ports:
      - "127.0.0.1:8000:3000"
      - "192.168.10.1:2222:22"
networks:
```

```
gitea:  
  external: false  
  driver_opts:  
    com.docker.network.bridge.name: br_gitea
```

Algunos puntos a destacar de esta configuración son:

- Se utilizará la imagen de Gitea de la versión **1.17.3** (la última a fecha de hoy).
- El **restart: always** indica al servicio de Docker que, si se para el contenedor, debe reiniciarlo.
- Se persiste el directorio del contenedor **/data** (donde se almacena la información guardada en Gitea) a la ruta del host **/srv/data/gitea/data** (esto nos será de utilidad para el proceso de backup) utilizando un volumen de Docker.
- Se mapea el puerto del servicio web que corre internamente el puerto 3000 del contenedor, al puerto 8000 de la dirección local de la máquina (al cual es posible llegar gracias a Nginx como proxy inverso). Por otro lado, se mapea el puerto SSH a la dirección del servidor VPN en el puerto 2222 (esto se explicará en la sección de clonación de proyectos por SSH).
- Finalmente, el bloque de la configuración de la red de "Gitea", se añade para que, al levantar el contenedor, cuando se listen las interfaces, la interfaz de red que levanta el contenedor aparezca con el nombre **br\_gitea** (esto es más por TOC que otra cosa).

#### Note

Para más detalles respecto al proceso de instalación con Docker (por ejemplo: para utilizar otras bases de datos, etc). Consultar la [documentación oficial](#)

Este archivo YAML, lo vamos a situar en la ruta: **/srv/containers/gitea/docker-compose.yml**. Previamente al despliegue del contenedor, se debe crear el directorio para guardar los volúmenes de la aplicación:

```
mkdir -p /srv/data/gitea/
```

Para proceder a su despliegue simplemente ejecutamos el siguiente comando:

```
docker compose up -d
```

## Note

Al especificar el flag `-d` le indicamos a docker compose que se despliegue el contenedor en "daemon mode", lo que nos impide ver la información del despliegue en tiempo real. Para poder acceder a este detalle simplemente ejecutamos:

```
docker compose logs -f
```

Tras el despliegue podemos ver como el contenedor esta corriendo con:

```
docker compose ps
```

```
root@vps:/srv/containers/gitea# ls
docker-compose.yml
root@vps:/srv/containers/gitea# docker compose up -d
[+] Running 9/9
  └─ gitea Pulled          41.4s
    └─ 213ec9aae27d Pull complete   1.8s
    └─ 6e6c408eba18 Pull complete   5.7s
    └─ 3b125bd762f1 Pull complete   5.7s
    └─ 83c809239a30 Pull complete   5.7s
    └─ 019ad2c6e82c Pull complete   31.1s
    └─ f725db2042da Pull complete   31.2s
    └─ 14a292704a22 Pull complete   32.2s
    └─ 7c415fa9989b Pull complete   32.2s
[+] Running 2/2
  └─ Network gitea_gitea Created      0.0s
  └─ Container gitea Started        0.9s
root@vps:/srv/containers/gitea# docker compose ps
NAME           COMMAND            SERVICE      STATUS        PORTS
gitea          "/usr/bin/entrypoint..."  gitea       running     192.168.10.1:2222->22/tcp, 127.0.0.1:8000->3000/tcp
root@vps:/srv/containers/gitea#
```

Tras el despliegue del contenedor, se procede a crear un virtual host en Nginx para poder hacer la redirección. Primero se define un subdominio al `/etc/hosts` para gestionar el servicio quedando así:

```
127.0.0.1 localhost
127.0.1.1 vps

# The following lines are desirable for IPv6 capable hosts
::1      ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

# LAB.LOCAL DNS records
192.168.10.1 vps.lab.local
192.168.10.2 laptop.lab.local

192.168.10.1 git.lab.local
```

También se crea el archivo de configuración de Nginx para gestionar el virtual host. Este archivo se sitúa en `/etc/nginx/sites-available/git.lab.local` y tiene el siguiente contenido:

```
server {  
    listen 192.168.10.1:80;  
    server_name git.lab.local;  
    # Redirect all non-https requests  
    rewrite ^ https://$host$request_uri? permanent;  
}  
  
server {  
    listen 192.168.10.1:443 ssl http2;  
  
    include snippets/self-signed.conf;  
    include snippets/ssl-params.conf;  
  
    server_name git.lab.local;  
  
    client_max_body_size 0;  
    proxy_http_version 1.1;  
    proxy_request_buffering off;  
  
    location / {  
        proxy_pass http://127.0.0.1:8000;  
        proxy_set_header Host $host;  
        proxy_set_header X-Real-IP $remote_addr;  
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
        proxy_set_header X-Forwarded-Proto $scheme;  
    }  
}
```

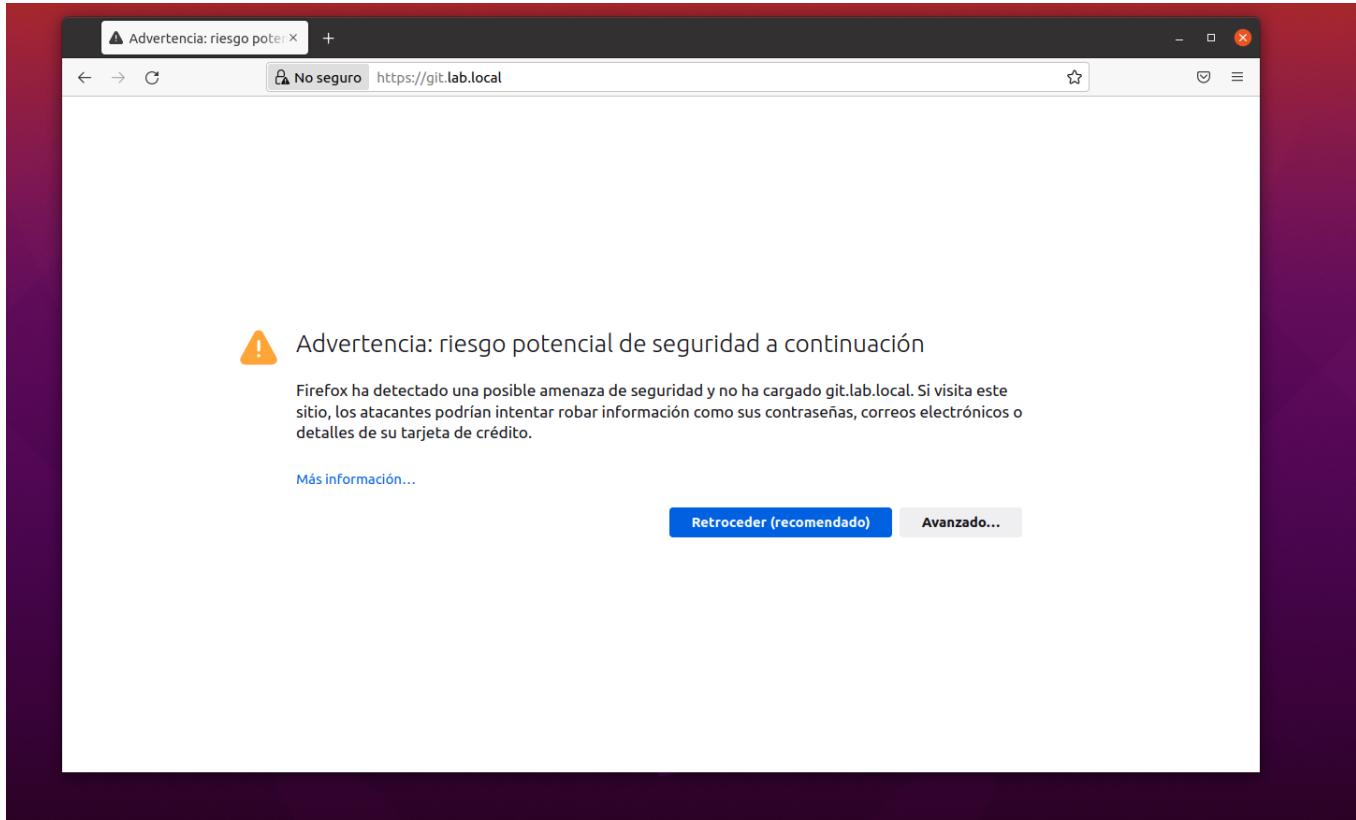
Tras esto, se crea el enlace simbólico para activarlo:

```
ln -s /etc/nginx/sites-available/git.lab.local /etc/nginx/sites-  
enabled/git.lab.local
```

Se comprueba que todo esta correcto y se reinicia el servicio de Nginx y Dnsmasq para aplicar los cambios:

```
nginx -t  
systemctl restart nginx  
systemctl restart dnsmasq
```

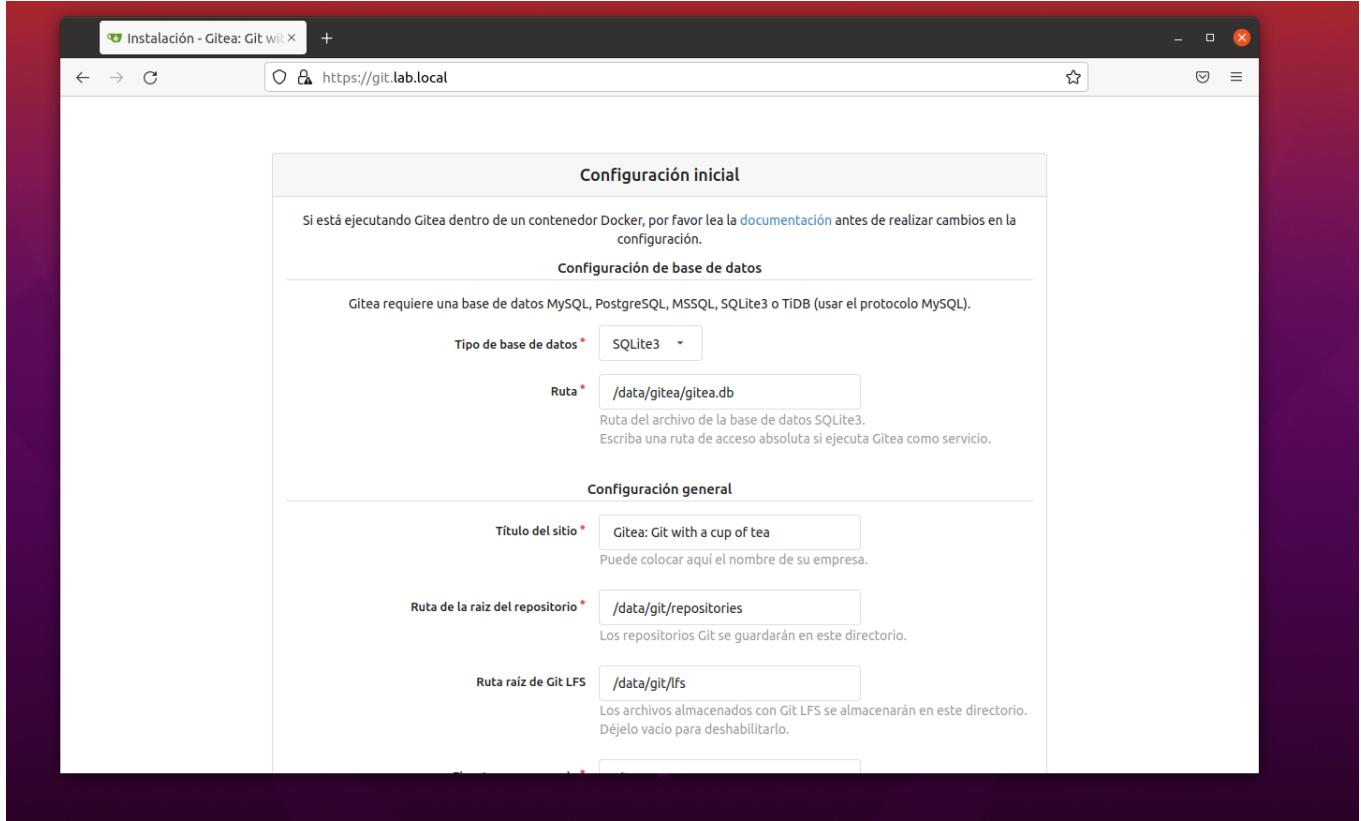
Y, si vamos a la máquina **LAPTOP** (cliente de Wireguard) y visitamos en un navegador la dirección <https://git.lab.local> nos encontramos con el siguiente resultado:



### Note

Esta advertencia se debe a que se está utilizando un certificado autofirmado en el Nginx (que hace de reverse proxy), simplemente tenemos que hacer click en **Avanzado...** y **Aceptar el riesgo y continuar.**

Tras esto nos encontramos con la página principal de Gitea donde se figura el proceso de configuración del Servicio:



## Configuración e instalación de Gitea

Continuando con la configuración de Gitea, nos situamos en la "Configuración Inicial" del paso anterior, dónde modificaremos los siguientes campos:

- **Título del sitio:** aquí simplemente se pondrá un título a gusto propio. En mi caso pondré: **Gitea: LAB.LOCAL**.
- **Dominio del servidor:** aquí se pondrá el valor **git-lablocal** (el cual tomará sentido en la sección de configuración de clonaciones por SSH).
- **URL Base de Gitea:** aquí lo dejamos con la URL que hemos utilizado para acceder. En mi caso: **https://git.lab.local/**.

Más adelante, en la configuración opcional, dejamos el siguiente estado:

Configuración opcional

► Configuración de Correo

▼ Configuración del servidor y de servicios de terceros

**Habilitar autenticación Local**

**Desactivar Gravatar**

**Habilitar avatares federados**

**Habilitar el inicio de sesión con OpenID**

**Deshabilitar auto-registro**

**Permitir el registro únicamente a través de servicios externos**

**Habilitar el auto-registro con OpenID**

**Requerir CAPTCHA durante el registro**

**Requerir inicio de sesión para ver páginas**

**Ocultar direcciones de correo electrónico por defecto**

**Permitir la creación de organizaciones por defecto**

**Activar el seguimiento de tiempo por defecto**

Dominio de correos electrónicos ocultos

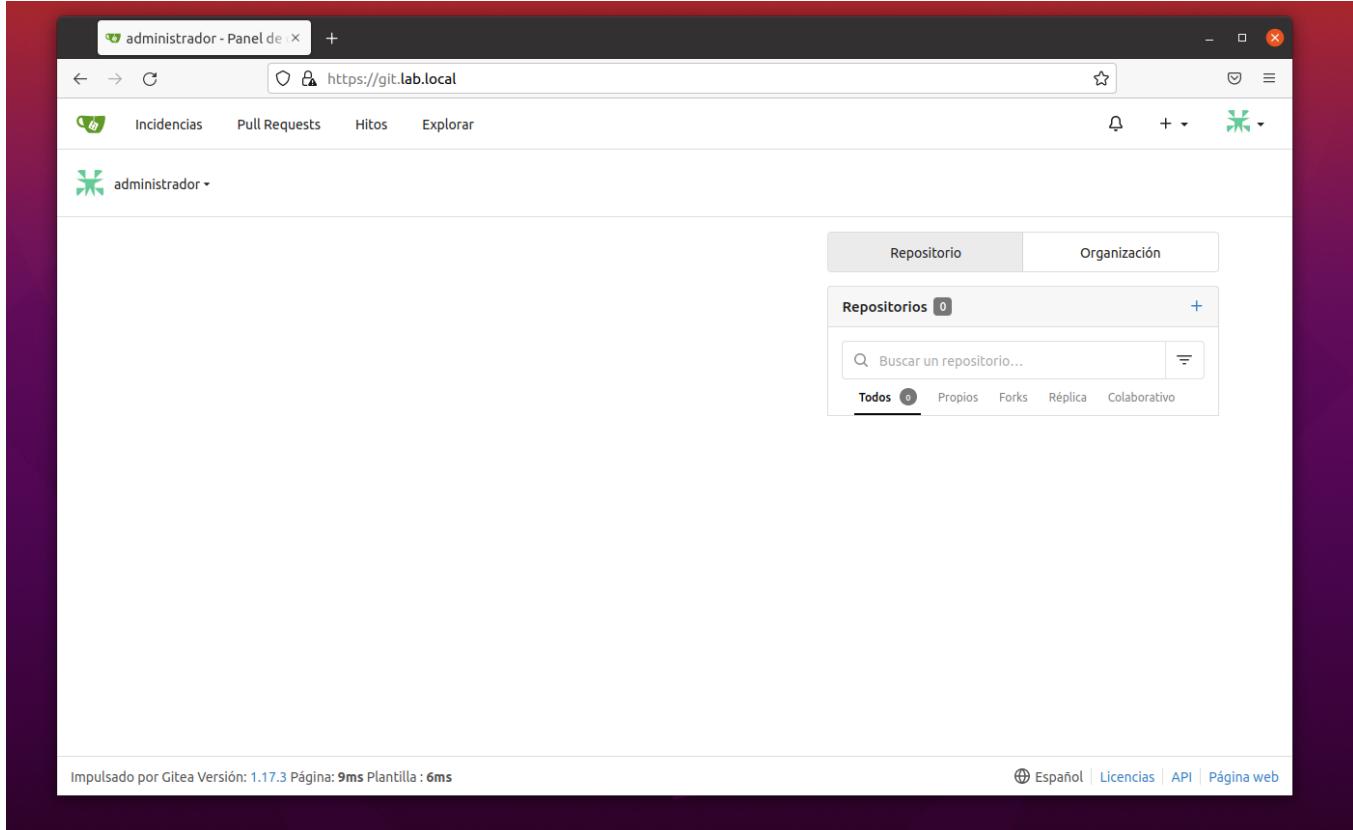
Nombre de dominio para usuarios con dirección de correo electrónico oculta. Por ejemplo, el usuario 'joe' quedará registrado en Git como 'joe@noreply.example.org' si el dominio de correo electrónico oculto se establece a 'noreply.example.org'.

Algoritmo Hash de Contraseña  ▾

Finalmente en la sección de **Configuración de la cuenta del administrador**, se configura el acceso a la cuenta de admin para gestionar Gitea:

- **Nombre del usuario:** administrador
- **Contraseña:** Sup3rAdm1nP4ssw0rd!
- **Correo Electrónico:** [administrador@lab.local](mailto:administrador@lab.local)

Tras esto se obtiene acceso al panel principal de Gitea con el usuario Administrador:



## Organización y gestión de Gitea

Tras haber concluido con la configuración inicial, se procede con la gestión de Gitea.

El primer paso consiste en la **Creación de una Organización**. Las organizaciones en Gitea permiten agrupar repositorios dándoles un cierto sentido grupal. Un ejemplo de uso real de una organización sería el de "simular" la *división departamental* dentro de una infraestructura empresarial.

### Note

Por ejemplo, el Equipo de **Red Team** tiene una organización sobre la que dispone de proyectos internos enfocados a la propia actividad del Red Team. El Equipo de **Blue Team** también dispondría de una organización propia con sus proyectos propios.

Nueva organización

Nombre de la organización \* RED-TEAM  
Los nombres de organización deben ser cortos y destacados.

Visibilidad \*  Público  
 Limitado (Visible sólo para los usuarios registrados)  
 Privado (Visible sólo para miembros de la organización)

Permisos  El administrador del repositorio puede añadir y eliminar el acceso a equipos

[Crear Organización](#) [Cancelar](#)

Impulsado por Gitea Versión: 1.17.3 Página: 1ms Plantilla: 0ms [Español](#) | [Licencias](#) | [API](#) | [Página web](#)

Continuando el proceso de gestión, el segundo paso sería la **Creación de un usuario de Gitea**. Esto se puede hacer desde la sección de "Administración del Sitio" > "Cuentas de Usuario".

Gestión de cuentas de usuario (Total: 1) [Crear Cuenta de Usuario](#)

| ID | Nombre de usuario | Correo electrónico      | Activado | Administrador | Restringido | 2FA | Repositorios | Creado       | Último registro  | Editar                 |
|----|-------------------|-------------------------|----------|---------------|-------------|-----|--------------|--------------|------------------|------------------------|
| 1  | administrador     | administrador@lab.local | ✓        | ✓             | ✗           | ✗   | 0            | Oct 16, 2022 | Nunca registrado | <a href="#">editar</a> |

Impulsado por Gitea Versión: 1.17.3 Página: 2ms Plantilla: 1ms [Español](#) | [Licencias](#) | [API](#) | [Página web](#)

Crear Cuenta de Usuario

Origen de Autenticación \* Local

Visibilidad del usuario \* Privado

Nombre de usuario \*

secu

Correo electrónico \*

secu@lab.local

Contraseña \*

.....

Obligar al usuario a cambiar la contraseña (recomendado)

Crear Cuenta de Usuario

Impulsado por Gitea Versión: 1.17.3 Página: 1ms Plantilla : 1ms

Español | Licencias | API | Página web

Tras crear al usuario, se le puede añadir a la nueva organización creada:

Owners - Gitea: LAB.LOCAL

RED-TEAM Privado

Equipos

Owners

Este equipo no tiene descripción

Los propietarios tienen acceso total a todos los repositorios y tienen acceso administrativo a la organización.

Configuración

2 miembros 0 repositorios

Buscar usuario... Añadir miembro al equipo

administrador

secu

Configuración

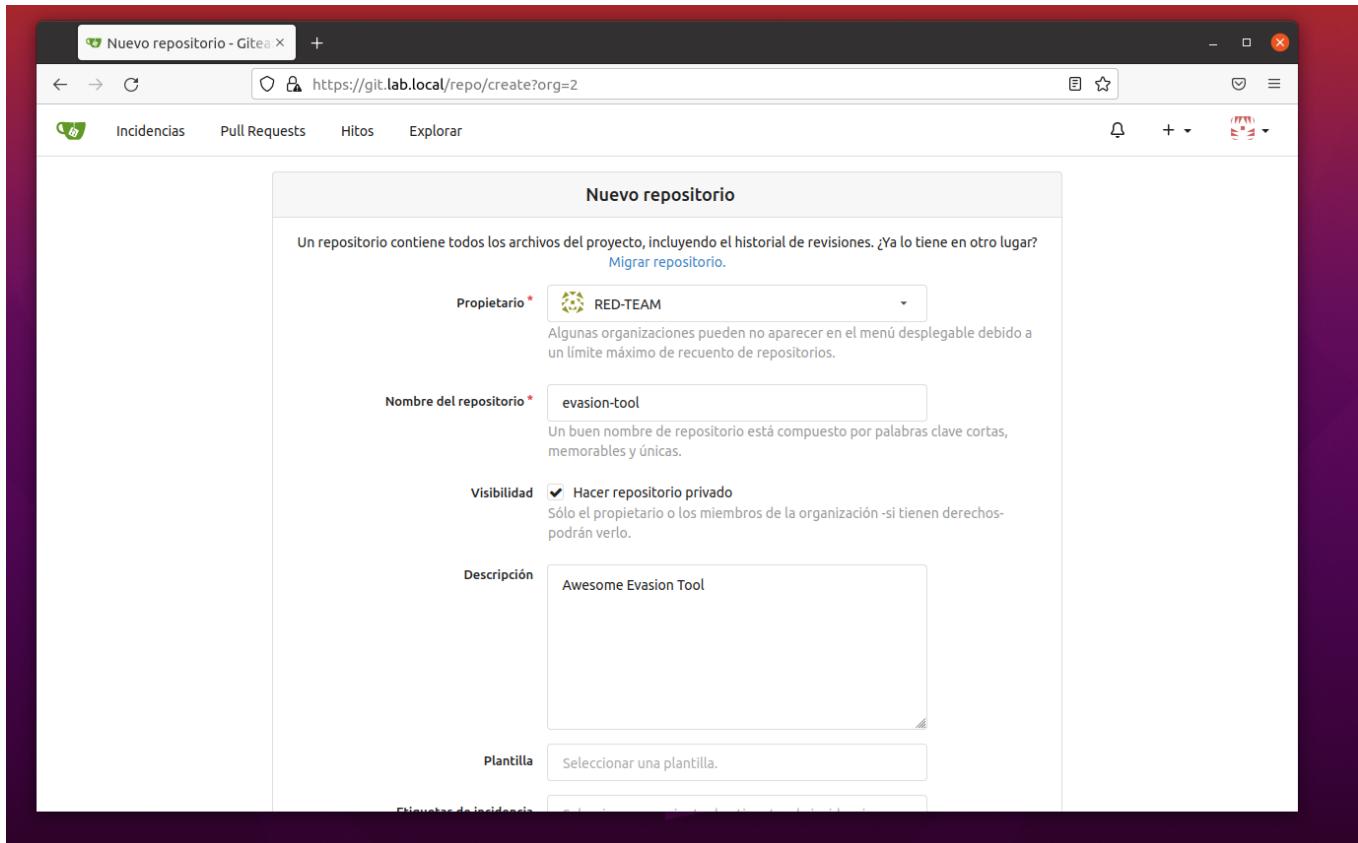
Impulsado por Gitea Versión: 1.17.3 Página: 2ms Plantilla : 1ms

Español | Licencias | API | Página web

Una vez se dispone de una Organización y una cuenta de Usuario, se puede comenzar con la creación de un repositorio. Para este proceso utilizaremos la cuenta local creada

en el paso anterior: @secu.

Para este caso, crearemos un repositorio dentro de la organización, pero también es posible crear repositorios pertenecientes al propio usuario.

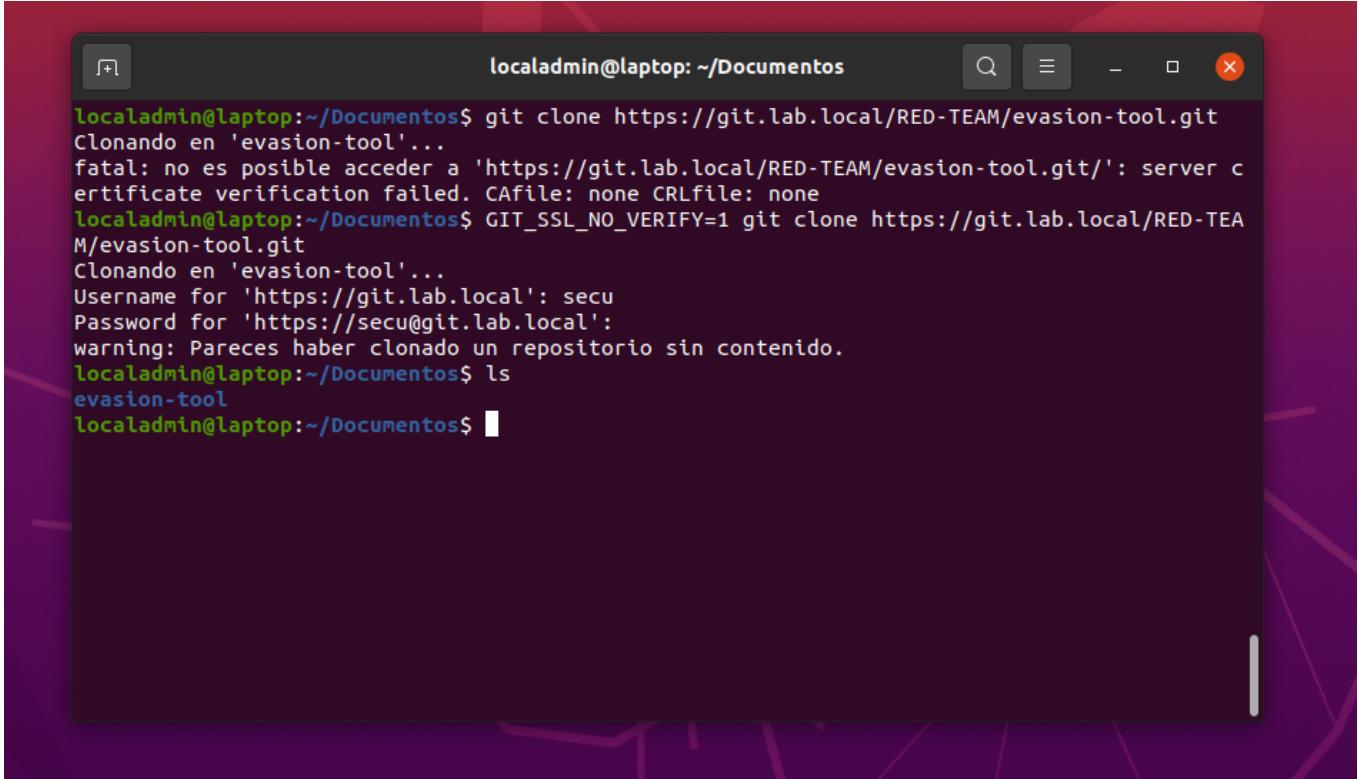


Tras crear el repositorio, se puede clonar vía HTTPS utilizando las credenciales del usuario secu. Previamente a la clonación se instala git:

```
apt install -y git
```

Tras esto, desde el directorio de la máquina **LAPTOP** que nos interese, se ejecuta:

```
GIT_SSL_NO_VERIFY=1 git clone https://git.lab.local/RED-TEAM/evasion-tool.git
```



```
localadmin@laptop:~/Documentos$ git clone https://git.lab.local/RED-TEAM/evasion-tool.git
Clonando en 'evasion-tool'...
fatal: no es posible acceder a 'https://git.lab.local/RED-TEAM/evasion-tool.git': server certificate verification failed. CAfile: none CRLfile: none
localadmin@laptop:~/Documentos$ GIT_SSL_NO_VERIFY=1 git clone https://git.lab.local/RED-TEAM/evasion-tool.git
Clonando en 'evasion-tool'...
Username for 'https://git.lab.local': secu
Password for 'https://secu@git.lab.local':
warning: Pareces haber clonado un repositorio sin contenido.
localadmin@laptop:~/Documentos$ ls
evasion-tool
localadmin@laptop:~/Documentos$
```

### Note

Es necesario pasar la variable de entorno `GIT_SSL_NO_VERIFY=1` a git, debido a que, al utilizar Nginx un certificado autofirmado, git no es capaz de validar el certificado, por tanto, se tiene que especificar que no proceda a validararlo. También es posible hacer una definición global de esta variable para no tener que especificarla siempre, no obstante, se recomienda utilizar la clonación por SSH en lugar de HTTPS (que se explicará a continuación).

## Habilitando clonaciones vía SSH

El último paso relativo a Gitea es configurar las clonaciones vía SSH en lugar de utilizar HTTPS. Las clonaciones por SSH nos evita tener que autenticarnos con las credenciales del usuario en cuestión. Lo que resulta bastante cómodo y, en cierto modo, seguro.

Para configurar las clonaciones por SSH se deben seguir los siguientes pasos:

1. Se genera una clave SSH con el comando: `ssh-keygen`. Tras esto encontraremos en el directorio `~/.ssh/` la clave pública (`~/.ssh/id_rsa.pub`) y la clave privada (`~/.ssh/id_rsa`).
2. Se copia el contenido de la clave pública a nuestro perfil de Gitea, en la sección de claves SSH.

The screenshot shows the Gitea configuration interface for managing SSH keys. The top navigation bar includes links for Incidencias, Pull Requests, Hitos, Explorar, Perfil, Cuenta, Apariencia, Seguridad, Aplicaciones, SSH / claves GPG (which is selected), Organizaciones, and Repositorios. The main content area is titled "Gestionar Claves SSH". It contains a form for adding a new key, with fields for "Nombre de la Clave" (set to "localadmin@laptop") and "Contenido" (containing a long SSH public key). Below the form are buttons for "Añadir Clave" and "Cancelar". A note below the form states: "Estas claves públicas SSH están asociadas con su cuenta. Las correspondientes claves privadas permiten acceso completo a sus repositorios." Another section titled "Administrador claves GPG" is also visible, with a note stating: "Estas claves públicas GPG están asociadas con su cuenta. Mantenga sus claves privadas a salvo, ya que permiten verificar commits." A link at the bottom left points to GitHub's help page for SSH keys.

3. Se añade el siguiente contenido al archivo `~/.ssh/config` con el siguiente contenido:

```
Host git-lablocal
  Hostname 192.168.10.1
  Port 2222
  User git
  PubKeyAuthentication yes
```

### Note

Algunos campos de interés son:

- El nombre del host (`git-lablocal`) el cual es el nombre del host que le hemos puesto al contenedor del Gitea en la configuración inicial
- El hostname (con la dirección IP de la VPN)
- El puerto (`2222` que habíamos mapeado del docker-compose)

4. Tras esto, se puede proceder a clonar este repositorio con el siguiente comando:

```
git clone git@git-lablocal:RED-TEAM/evasion-tool.git
```

The screenshot shows a Gitea repository page for 'RED-TEAM/evasion-tool'. The top navigation bar includes links for 'Incidencias', 'Pull Requests', 'Hitos', and 'Explorar'. The main content area has tabs for 'Código', 'Incidencias', 'Paquetes', 'Proyectos', and 'Wiki'. A 'Guía rápida' sidebar is on the left. The central part displays cloning instructions and a terminal window. The terminal window shows the following session:

```
localadmin@laptop:~/Documentos$ git clone git@git-lablocal:RED-TEAM/evasion-tool.git
Clonando en 'evasion-tool'...
The authenticity of host '[192.168.10.1]:2222 ([192.168.10.1]:2222)' can't be established.
ECDSA key fingerprint is SHA256:uita25BcQmY4irv9s9Z1jt081jEt7dT96SWAQqhVngI.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[192.168.10.1]:2222' (ECDSA) to the list of known hosts.
warning: Pareces haber clonado un repositorio sin contenido.
localadmin@laptop:~/Documentos$ ls
evasion-tool
localadmin@laptop:~/Documentos$
```

Below the terminal window, there's a section titled 'Hacer push de un repositorio' with the command:

```
git remote add origin git@git-lablocal:RED-TEAM/evasion-tool.git
git push -u origin main
```

Y es por esto por la razón que se le ha puesto como **Dominio del servidor** = `git-lablocal` en la configuración inicial. Pues, de esta forma, en la pestaña de clonar repositorio por **SSH** el propio Gitea nos da la dirección exacta de git que tenemos que utilizar para hacer el clone vía SSH a través de la VPN.

## Servicio de Edición Colaborativo - Hackmd

The image features the HackMD logo, which consists of a white icon of a document with horizontal lines and the word 'HackMD' in a large, bold, white sans-serif font. Below the logo, the tagline 'Get everyone on the same page with Markdown' is written in a smaller, white, sans-serif font.

## ¿Por qué utilizar Hackmd?



Hackmd es un editor colaborativo en tiempo real que se utiliza para documentación técnica en Markdown - extracto de la [web oficial de Hackmd](#).

Desde mi punto de vista Hackmd es muy útil en una organización o infraestructura pues, permite trabajar varias personas a la vez sobre un mismo documento simultáneamente. Esta capacidad permite ponerse de acuerdo sobre: ideas, proyectos, documentaciones, o cualquier tarea de coordinación.

No conozco otro servicio similar, ni tampoco requiere de grandes requerimientos para funcionar, su despliegue es fácil y rápido, así que eso lo convierte en un software idóneo a tener en la infraestructura.

## Despliegue de contenedor de Hackmd

Para desplegar el contendor de Hackmd de forma automatizada se utilizará el siguiente archivo YAML de Docker-Compose:

```
version: "3"
services:
  database:
    image: postgres:11.6-alpine
    environment:
      - POSTGRES_USER=codimd
      - POSTGRES_PASSWORD=Postgr3sP4ssw0rd
      - POSTGRES_DB=codimd
    volumes:
      - "/srv/data/hackmd/database-data:/var/lib/postgresql/data"
    restart: always
    ports:
      - "127.0.0.1:5432:5432"
    networks:
      - hackmd
  codimd:
    image: hackmdio/hackmd:2.4.2
    environment:
      - CMD_DB_URL=postgres://codimd:Postgr3sP4ssw0rd@database/codimd
      - CMD_USECDN=false
    depends_on:
      - database
```

```
ports:
  - "127.0.0.1:8001:3000"
volumes:
  - "/srv/data/hackmd/upload-data:/home/hackmd/app/public/uploads"
restart: always
networks:
  - hackmd
networks:
  hackmd:
    driver_opts:
      com.docker.network.bridge.name: br_hackmd
```

Algunos puntos a destacar de esta configuración son:

- A diferencia con el archivo de configuración de docker-compose de Gitea, en este se define la creación de dos contenedores: database (postgres 11.6) y codimd (hackmd 2.4.2). Ambos conectados por la network "hackmd".
- En el campo **environment** de cada contenedor, se definen un conjunto de variables de entorno que corresponden a credenciales y otra información necesaria para el despliegue del contenedor (esta información debe modificarse).
- El **depends\_on** del contenedor de codimd, indica que este contenedor depende de **database** (el cual se refiere al contenedor de la base de datos) y esto tiene sentido pues, sin el contenedor de la base de datos, no se puede levantar la aplicación de Hackmd. A lo que esto se traduce en el despliegue es que, primero se levantará el contenedor de la base de datos y, posteriormente, se hará el de hackmd.

#### Note

Para más detalles respecto al proceso de instalación con Docker, consultar la [documentación oficial](#)

Este archivo YAML, lo vamos a situar en la ruta: `/srv/containers/hackmd/docker-compose.yml`. Antes de levantar el contenedor, se crea el directorio de datos para los volúmenes:

```
mkdir -p /srv/data/hackmd/
```

Para proceder a su despliegue simplemente ejecutamos el siguiente comando:

```
docker compose up -d
```

Tras el despliegue podemos ver como el contenedor esta corriendo con:

```
docker compose ps
```

```
root@vps:/srv/containers/hackmd# docker compose up -d
[+] Running 20/20
  # codimd Pulled
  # f7ec5a41d630 Pull complete          88.8s
  # af85e22911d9 Pull complete          20.5s
  # e9aab8c581ea Pull complete          20.5s
  # 64094dc84c04 Pull complete          33.4s
  # cdbc69d5eb3c Pull complete          33.5s
  # 3af76c12308f Pull complete          33.6s
  # 4f4fb700ef54 Pull complete          56.6s
  # elf2bc4abf44 Pull complete          56.7s
  # 0d4e1285fef2 Pull complete          79.4s
  # database Pulled                   85.0s
  # c9b1b535fdd9 Pull complete          20.2s
  # d1030c456d04 Pull complete          1.8s
  # d1d0211bd9a Pull complete          1.8s
  # 07d0560c0a3f Pull complete          1.9s
  # ce7fd4584a5f Pull complete          16.4s
  # 63eb0325fe1c Pull complete          16.4s
  # b67486507716 Pull complete          16.4s
  # f58de2b85820 Pull complete          16.5s
  # ca982626dd56 Pull complete          16.5s
[+] Running 3/3
  # Network hackmd_hackmd      Created           0.0s
  # Container hackmd-database-1 Started          1.4s
  # Container hackmd-codimd-1 Started          0.6s
root@vps:/srv/containers/hackmd# docker compose ps
             COMMAND            SERVICE    STATUS        PORTS
NAME                  COMMAND            SERVICE    STATUS        PORTS
hackmd-codimd-1   "/home/hackmd/app/do..."  codimd    running
hackmd-database-1  "docker-entrypoint.s..."  database  running
127.0.0.1:8001->3000/tcp
127.0.0.1:5432->5432/tcp
root@vps:/srv/containers/hackmd#
```

Se añade el subdominio al `/etc/hosts` para gestionar el servicio (como se ha hecho anteriormente) quedando el archivo así:

```
127.0.0.1 localhost
127.0.1.1 vps

# The following lines are desirable for IPv6 capable hosts
::1      ip6-localhost ip6-loopback
fe00::0  ip6-localnet
ff00::0  ip6-mcastprefix
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters

# LAB.LOCAL DNS records
192.168.10.1 vps.lab.local
192.168.10.2 laptop.lab.local

192.168.10.1 git.lab.local
192.168.10.1 md.lab.local
```

También se crea el archivo de configuración de Nginx para gestionar el virtual host. Este archivo se situa en `/etc/nginx/sites-available/md.lab.local` y tiene el siguiente contenido:

```

server {
    listen 192.168.10.1:80;
    server_name md.lab.local;
    # Redirect all non-https requests
    rewrite ^ https://$host$request_uri? permanent;
}

server {
    listen 192.168.10.1:443 ssl http2;

    include snippets/self-signed.conf;
    include snippets/ssl-params.conf;

    server_name md.lab.local;

    client_max_body_size 0;
    proxy_http_version 1.1;
    proxy_request_buffering off;

    location / {
        proxy_pass http://127.0.0.1:8001;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

```

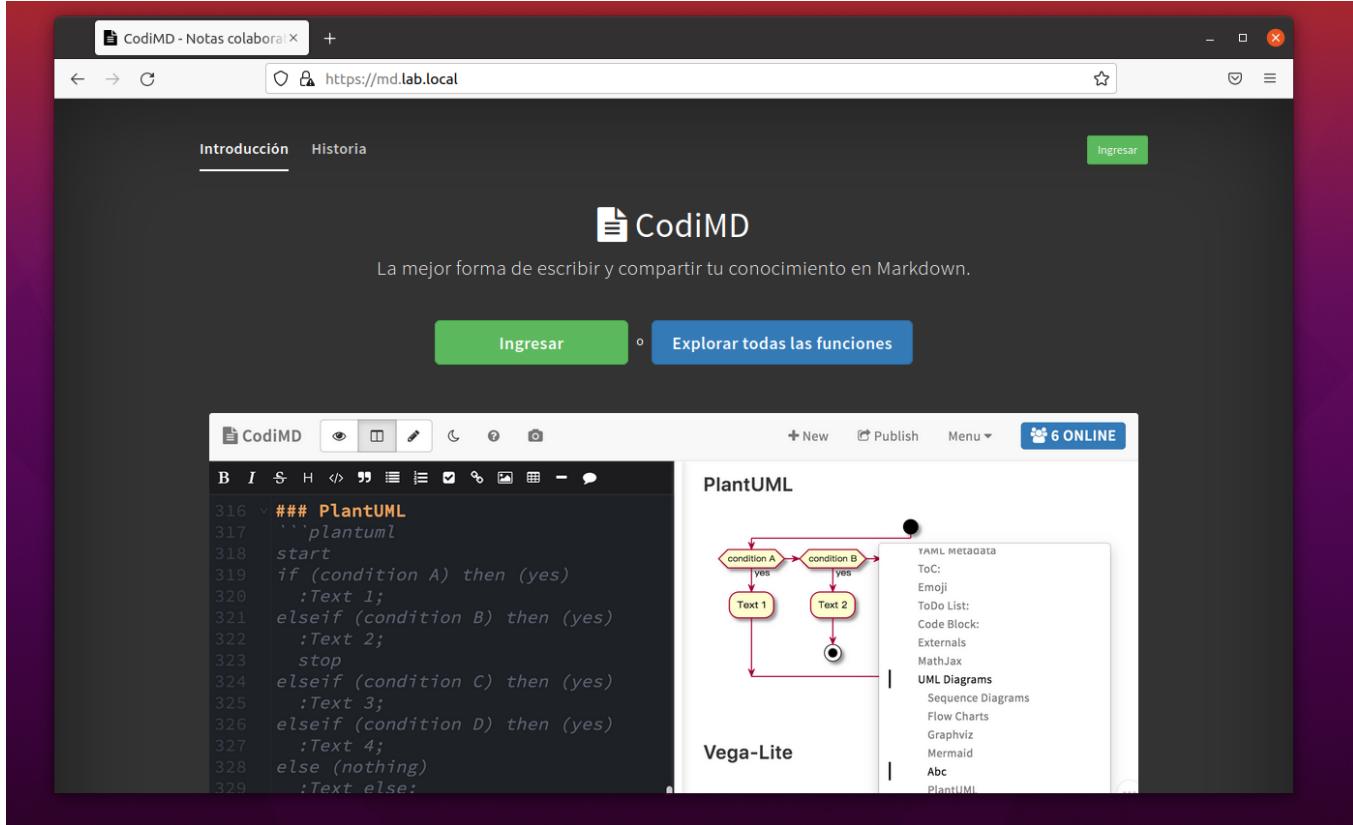
Tras esto, se crea el enlace simbólico para activarlo:

```
ln -s /etc/nginx/sites-available/md.lab.local /etc/nginx/sites-enabled/md.lab.local
```

Se comprueba que todo esta correcto y se reinicia el servicio de Nginx y Dnsmasq para aplicar los cambios:

```
nginx -t
systemctl restart nginx
systemctl restart dnsmasq
```

Y, desde la máquina **LAPTOP**, visitamos la dirección <https://md.lab.local> (tras excepcionar el certificado SSL) encontramos:

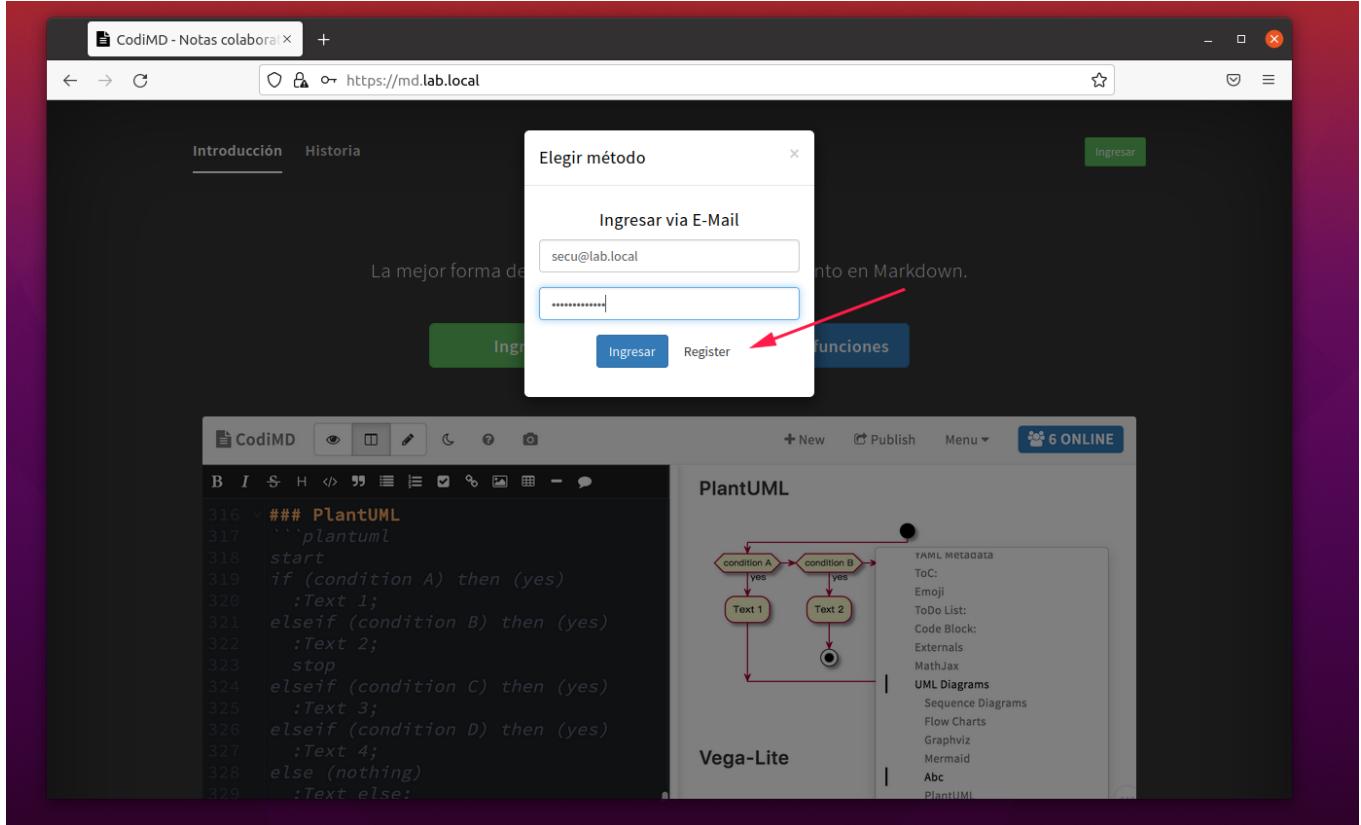


## Configuración y gestión de Hackmd

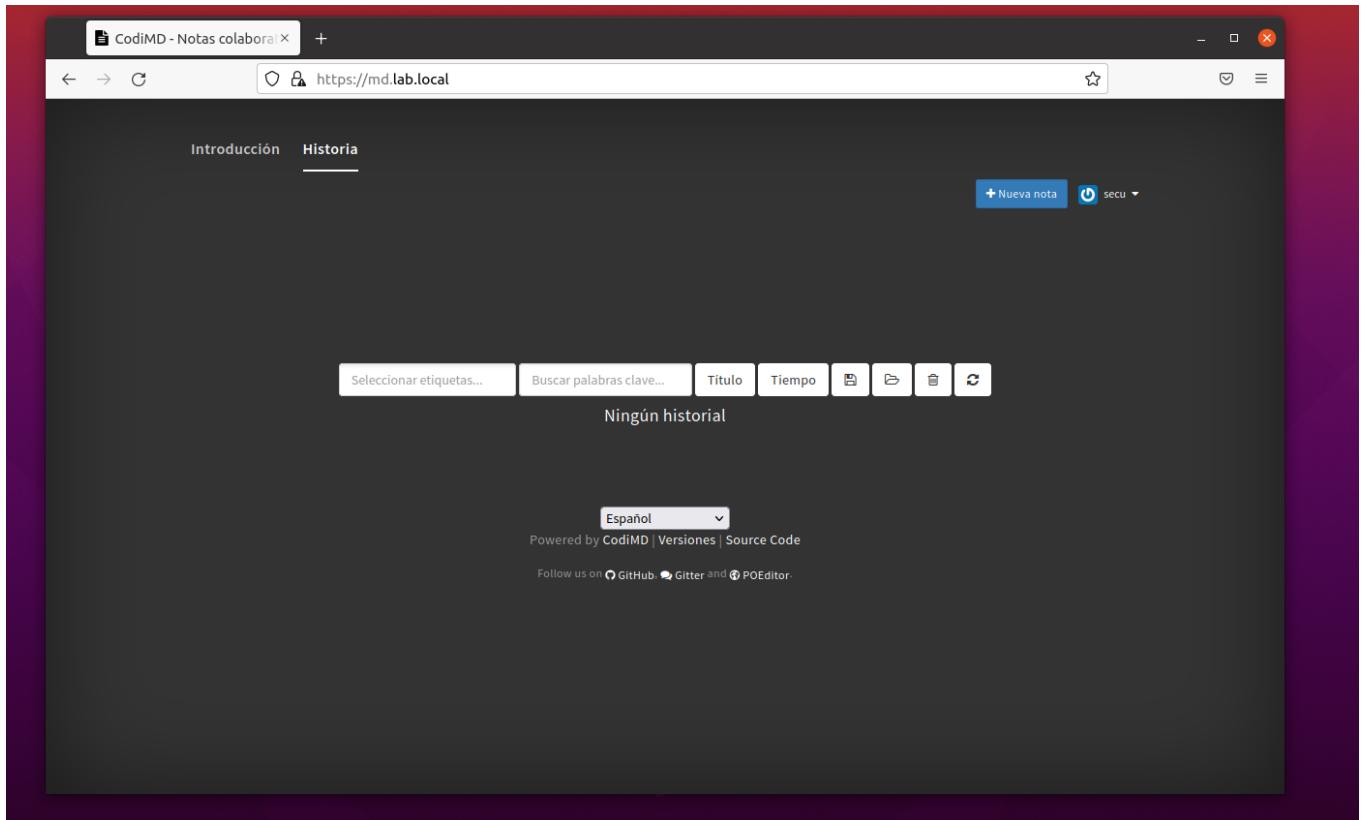
La configuración que vamos a realizar en Hackmd es muy sencilla y directa, debido a que será lo mínimo que necesitaremos para poder utilizar correctamente Hackmd.

En primer lugar procederemos a crear un usuario haciendo click en el botón verde **Ingresar** (situado en la esquina superior derecha). Tras esto se abre un desplegable donde ponemos un correo y una contraseña.

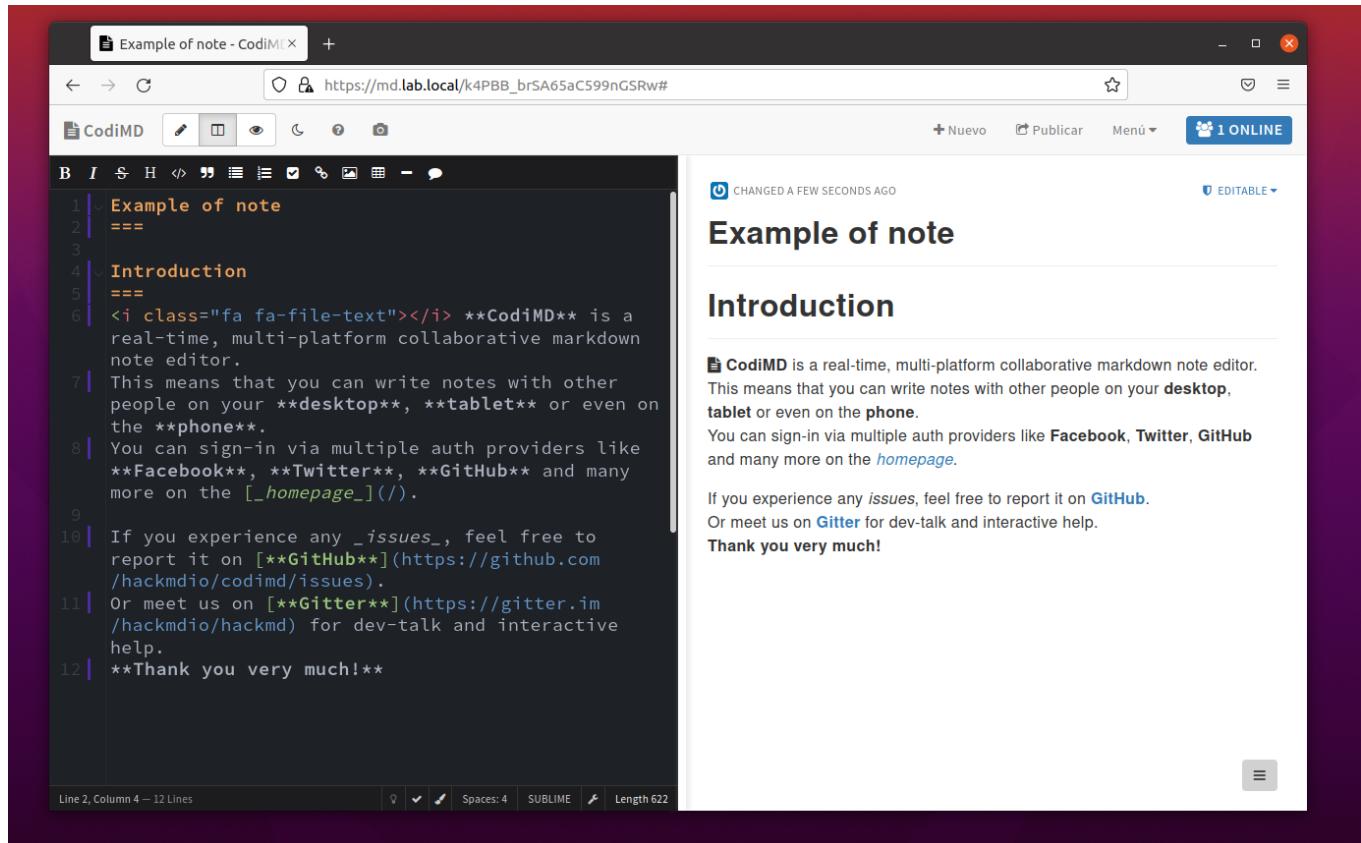
Al introducir estos dos datos, haremos click en el botón **Register** (como se ve en la siguiente imagen) para registrar al usuario.



Tras el registro, se inicia sesión con el correo y las credenciales introducidas y se redirige a la página principal del Hackmd.



En esta sección aparecen las notas creadas, al no existir ninguna, se procede a crear una.



Simplemente se destaca un bug que ocurre al hacer el despliegue con Docker, y es que, al intentar subir una imagen al documento, esta no aparece y, en el backend, ocurre un error:

### Note

Para ver los logs del backend, desde el directorio `/srv/containers/hackmd/` ejecutamos el comando: `docker compose logs -f`

```

Regular.woff?V=2.7.5 HTTP/1.1" 200 5476 "https://md.lab.local/features?both" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:105.0) Gecko/20100101 Firefox/105.0"
hackmd-codimd-1 | 2022-10-17T21:36:11.924Z info: User doesn't have the right to edit.
hackmd-codimd-1 | 2022-10-17T21:36:12.083Z info: User doesn't have the right to edit.
hackmd-codimd-1 | 2022-10-17T21:36:12.177Z info: User doesn't have the right to edit.
hackmd-codimd-1 | 2022-10-17T21:36:12.259Z info: User doesn't have the right to edit.
hackmd-codimd-1 | 2022-10-17T21:36:12.343Z info: User doesn't have the right to edit.
hackmd-codimd-1 | 2022-10-17T21:36:18.169Z info: User doesn't have the right to edit.
hackmd-codimd-1 | 2022-10-17T21:36:18.311Z info: User doesn't have the right to edit.
hackmd-codimd-1 | 2022-10-17T21:36:18.358Z info: User doesn't have the right to edit.
hackmd-codimd-1 | 2022-10-17T21:36:18.382Z info: User doesn't have the right to edit.
hackmd-codimd-1 | 2022-10-17T21:36:18.448Z info: User doesn't have the right to edit.
hackmd-codimd-1 | 2022-10-17T21:36:18.519Z info: User doesn't have the right to edit.
hackmd-codimd-1 | 2022-10-17T21:36:18.583Z info: User doesn't have the right to edit.
hackmd-codimd-1 | 2022-10-17T21:36:18.651Z info: User doesn't have the right to edit.
hackmd-codimd-1 | 2022-10-17T21:36:18.721Z info: User doesn't have the right to edit.
hackmd-codimd-1 | 2022-10-17T21:36:18.795Z info: User doesn't have the right to edit.
hackmd-codimd-1 | 2022-10-17T21:36:18.866Z info: User doesn't have the right to edit.
hackmd-codimd-1 | 2022-10-17T21:36:18.963Z info: User doesn't have the right to edit.
hackmd-codimd-1 | 2022-10-17T21:36:19.055Z info: User doesn't have the right to edit.
hackmd-codimd-1 | 2022-10-17T21:36:19.195Z info: User doesn't have the right to edit.
hackmd-codimd-1 | 2022-10-17T21:36:19.720Z info: User doesn't have the right to edit.
hackmd-codimd-1 | 2022-10-17T21:41:55.208Z info: deserializeUser: 924cfeba-5ec9-4bcc-9ee6-4eef79d55529
hackmd-codimd-1 | 2022-10-17T21:41:56.369Z info: deserializeUser: 924cfeba-5ec9-4bcc-9ee6-4eef79d55529
hackmd-codimd-1 | 2022-10-17T21:41:56.372Z info: 172.19.0.1 - - [17/Oct/2022:21:41:56 +0000] "GET /me HTTP/1.1" 200 153 "https://md.lab.local/k4PBB_bri$A65aC599nGSRw" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:105.0) Gecko/20100101 Firefox/105.0"
hackmd-codimd-1 |
hackmd-codimd-1 | 2022-10-17T21:42:06.703Z info: deserializeUser: 924cfeba-5ec9-4bcc-9ee6-4eef79d55529
hackmd-codimd-1 | 2022-10-17T21:42:06.715Z error: EACCES: permission denied, copyfile '/tmp/upload_5365168f93fed73233ebcc655e77794a.png' -> '/home/hackmd/app/public/uploads/upload_5365168f93fed73233ebcc655e77794a.png'
hackmd-codimd-1 | 2022-10-17T21:42:06.722Z info: 172.19.0.1 - - [17/Oct/2022:21:42:06 +0000] "POST /uploadimage HTTP/1.1" 500 - "https://md.lab.local/k4PBB_bri$A65aC599nGSRw?both" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:105.0) Gecko/20100101 Firefox/105.0"
hackmd-codimd-1 |

```

Este error es por falta de permisos en el directorio del contenedor:

`/home/hackmd/app/public/uploads/`. Cabe destacar que esta falta de permisos se debe a que, al utilizarse como volumen el directorio del host: `/srv/data/hackmd/upload-data/`, hereda los permisos, los cuales no permiten a un usuario que no sea root, escribir en el directorio.

Para solucionarlo simplemente ejecutamos el siguiente comando en el servidor VPS:

```
chmod 777 /srv/data/hackmd/upload-data/
```

Tras esto, se vuelve a subir una imagen arrastrandola al documento de Hackmd y funciona correctamente:

The screenshot shows two side-by-side views of the CodiMD interface. On the left, the code editor view displays a note with the following content:

```

1 Example of note
2 ===
3
4
5 Introduction
6 ===
7 <i class="fa fa-file-text"></i> **CodiMD** is a real-time, multi-
platform collaborative markdown note editor.
8 This means that you can write notes with other people on your
**desktop**, **tablet** or even on the **phone**.
9 You can sign-in via multiple auth providers like **Facebook**,
**Twitter** , **GitHub** and many more on the [_homepage_](/).
10
11 If you experience any _issues_, feel free to report it on
[**GitHub**](https://github.com/hackmdio/codimd/issues).
12 Or meet us on [**Gitter**](https://gitter.im/hackmdio/hackmd) for
dev-talk and interactive help.
13 **Thank you very much!**
14
15 

```

On the right, the preview view shows the note with the heading 'Example of note' and the 'Introduction' section. It also includes a note about GitHub issues and a link to Gitter. Below the preview is a small image of a purple Gengar Pokémon.

# Servicio de Mensajería - Zulip



# ZULIP

## ¿Por qué Zulip?

### » Quote

"Chat para equipos distribuidos. Zulip combina la inmediatez del chat en tiempo real con un modelo de hilos de correo electrónico. Con Zulip, puedes ponerte al día en las conversaciones importantes mientras ignoras las irrelevantes." - extracto de [la página oficial de Zulip](#)

Tal y como se describe en la documentación oficial, Zulip es un servicio de mensajería instantánea multiplataforma y que se puede desplegar on-premises. Es una plataforma desarrollada en Python (Flask), se puede

Aunque Zulip no es la única solución open source para esta propósito. Existen otros servicios conocidos como: Rocket Chat o Mattermost.

El motivo por el cuál he decidido apostar por Zulip es por los siguientes puntos:

- No consume excesivamente recursos (dentro de que es un servicio de mensajería instantánea, como base puede consumir unos 2GB aprox).
- Su despliegue con Docker es muy sencillo. Todo empaquetado en un archivo Docker-Compose y las instrucciones de despliegue no son muy enrevesadas.
- Cumple todas las funciones a esperar de un servicio de mensajería y no se echa nada en falta.
- El sistema de Topics y Streams permite llevar una organización precisa y aislada de las tareas/actividades a realizar dentro de una organización.
- Dispone de aplicación de escritorio y móvil, que funciona bastante bien. Y en el tiempo que he utilizado el servicio no he podido apreciar fallos graves.

Obviamente, existirán soluciones que se adapten más a las necesidades de cada uno. no obstante, por las razones enumeradas anteriormente y por más que ya veremos a continuación yo me decanto por Zulip y, en las siguientes secciones comenzaremos con su despliegue, instalación y configuración.

## Despliegue de contenedor de Zulip

Empezaremos con el despliegue del contendor de Zulip de forma automatizada. Para ello se utilizará el siguiente archivo YAML usable con Docker-Compose:

```
version: "2"
services:
  database:
    image: "zulip/zulip-postgresql:10"
    environment:
      POSTGRES_DB: "zulip"
      POSTGRES_USER: "zulip"
      # Note that you need to do a manual `ALTER ROLE` query if you
      # change this on a system after booting the postgres container
      # the first time on a host. Instructions are available in README.md.
      POSTGRES_PASSWORD: "PostgresSecurePassword!"
    volumes:
      - "/srv/data/zulip/postgresql/data:/var/lib/postgresql/data:rw"
  memcached:
    image: "memcached:alpine"
    command:
      - "sh"
      - "-euc"
      - |
        echo 'mech_list: plain' > "$$SASL_CONF_PATH"
        echo "zulip@$$HOSTNAME:$$MEMCACHED_PASSWORD" >
        $$MEMCACHED_SASL_PWDB
        echo "zulip@localhost:$$MEMCACHED_PASSWORD" >>
        $$MEMCACHED_SASL_PWDB
        exec memcached -S
    environment:
      SASL_CONF_PATH: "/home/memcache/memcached.conf"
      MEMCACHED_SASL_PWDB: "/home/memcache/memcached-sasl-db"
      MEMCACHED_PASSWORD: "MemcachedSecurePassword!"
    restart: always
  rabbitmq:
    image: "rabbitmq:3.7.7"
    restart: always
    environment:
      RABBITMQ_DEFAULT_USER: "zulip"
```

```
RABBITMQ_DEFAULT_PASS: "RabbitmqSecurePassword!"  
volumes:  
  - "/srv/data/zulip/rabbitmq:/var/lib/rabbitmq:rw"  
redis:  
  image: "redis:alpine"  
  command:  
    - "sh"  
    - "-euc"  
    - |  
      echo "requirepass $$REDIS_PASSWORD" > /etc/redis.conf  
      exec redis-server /etc/redis.conf  
environment:  
  REDIS_PASSWORD: "RedisSecurePassword!"  
volumes:  
  - "/srv/data/zulip/redis:/data:rw"  
zulip:  
  image: "zulip/docker-zulip:5.6-0"  
  build:  
    context: .  
    args:  
      # Change these if you want to build zulip from a different  
      # repo/branch  
      ZULIP_GIT_URL: https://github.com/zulip/zulip.git  
      ZULIP_GIT_REF: "5.6"  
      # Set this up if you plan to use your own CA certificate bundle for  
      # building  
      # CUSTOM_CA_CERTIFICATES:  
  ports:  
    - "127.0.0.1:8002:80"  
    - "127.0.0.1:8443:443"  
environment:  
  DB_HOST: "database"  
  DB_HOST_PORT: "5432"  
  DB_USER: "zulip"  
  SSL_CERTIFICATE_GENERATION: "self-signed"  
  SETTING_MEMCACHED_LOCATION: "memcached:11211"  
  SETTING_RABBITMQ_HOST: "rabbitmq"  
  SETTING_REDIS_HOST: "redis"  
  SECRETS_email_password: "123456789"  
  # These should match RABBITMQ_DEFAULT_PASS, POSTGRES_PASSWORD,  
  # MEMCACHED_PASSWORD, and REDIS_PASSWORD above.  
  SECRETS_rabbitmq_password: "RabbitmqSecurePassword!"  
  SECRETS_postgres_password: "PostgresSecurePassword!"  
  SECRETS_memcached_password: "MemcachedSecurePassword!"  
  SECRETS_redis_password: "RedisSecurePassword!"  
  SECRETS_secret_key: "SecretSecureKey!"
```

```
SETTING_EXTERNAL_HOST: "chat.lab.local"
SETTING_ZULIP_ADMINISTRATOR: "admin@lab.local"
SETTING_EMAIL_HOST: "" # e.g. smtp.example.com
SETTING_EMAIL_HOST_USER: "noreply@lab.local"
SETTING_EMAIL_PORT: "587"
# It seems that the email server needs to use ssl or tls and can't be
used without it
SETTING_EMAIL_USE_SSL: "False"
SETTING_EMAIL_USE_TLS: "True"
ZULIP_AUTH_BACKENDS: "EmailAuthBackend"
# Uncomment this when configuring the mobile push notifications
service
# SETTING_PUSH_NOTIFICATION_BOUNCER_URL: 'https://push.zulipchat.com'
volumes:
- "/srv/data/zulip/zulip:/data:rw"
ulimits:
nofile:
soft: 1000000
hard: 1048576
```

### Note

Recuerda cambiar las contraseñas de los distintos servicios por unas suficientemente robustas. Las que aquí figuran son a modo de ejemplo para la guía.

Este archivo YAML, lo vamos a situar en la ruta: `/srv/containers/zulip/docker-compose.yml`. Antes de levantar el contenedor, se crea el directorio de datos para los volúmenes:

```
mkdir -p /srv/data/zulip/
```

Para proceder a su despliegue simplemente ejecutamos el siguiente comando (el cual tardará unos minutos entre que se descarga las imágenes y levanta los contenedores):

```
docker compose up -d
```

Tras el despliegue podemos ver como el contenedor esta corriendo con:

```
docker compose ps
```

```
root@vps:/srv/containers/zulip# docker compose ps
NAME          COMMAND           SERVICE      STATUS        PORTS
zulip-database-1 "docker-entrypoint.s..." database    running      5432/tcp
zulip-memcached-1 "docker-entrypoint.s..." memcached  running      11211/tcp
zulip-rabbitmq-1 "docker-entrypoint.s..." rabbitmq   running      4369/tcp, 5671-5672/tcp, 25672/tcp
zulip-redis-1   "docker-entrypoint.s..." redis      running      6379/tcp
zulip-zulip-1   "/sbin-entrypoint.sh"  zulip      running      127.0.0.1:8002->80/tcp, 127.0.0.1:8443->443/tcp
root@vps:/srv/containers/zulip#
```

Se añade el subdominio al `/etc/hosts` para gestionar el servicio (como se ha hecho anteriormente) quedando el archivo así:

```
127.0.0.1 localhost
127.0.1.1 vps

# The following lines are desirable for IPv6 capable hosts
::1      ip6-localhost ip6-loopback
fe00::0  ip6-localnet
ff00::0  ip6-mcastprefix
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters

# LAB.LOCAL DNS records
192.168.10.1 vps.lab.local
192.168.10.2 laptop.lab.local

192.168.10.1 git.lab.local
192.168.10.1 md.lab.local
192.168.10.1 chat.lab.local
```

También se crea el archivo de configuración de Nginx para gestionar el virtual host. Este archivo se sitúa en `/etc/nginx/sites-available/chat.lab.local` y tiene el siguiente contenido:

```
server {
    listen 192.168.10.1:80;
    server_name chat.lab.local;
    # Redirect all non-https requests
    rewrite ^ https://$host$request_uri? permanent;
}

server {
    listen 192.168.10.1:443 ssl http2;

    include snippets/self-signed.conf;
    include snippets/ssl-params.conf;

    server_name chat.lab.local;
```

```
client_max_body_size 0;  
proxy_http_version 1.1;  
proxy_request_buffering off;  
  
location / {  
    proxy_pass https://127.0.0.1:8443;  
    proxy_set_header Host $host;  
    proxy_set_header X-Real-IP $remote_addr;  
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
    proxy_set_header X-Forwarded-Proto $scheme;  
}  
}
```

### Note

En el archivo de configuración de Nginx se hace el `proxy_pass` al puerto HTTPS del Zulip debido a que, si se hace al puerto HTTP, se obtiene un bucle infinito de redirecciones al puerto HTTPS.

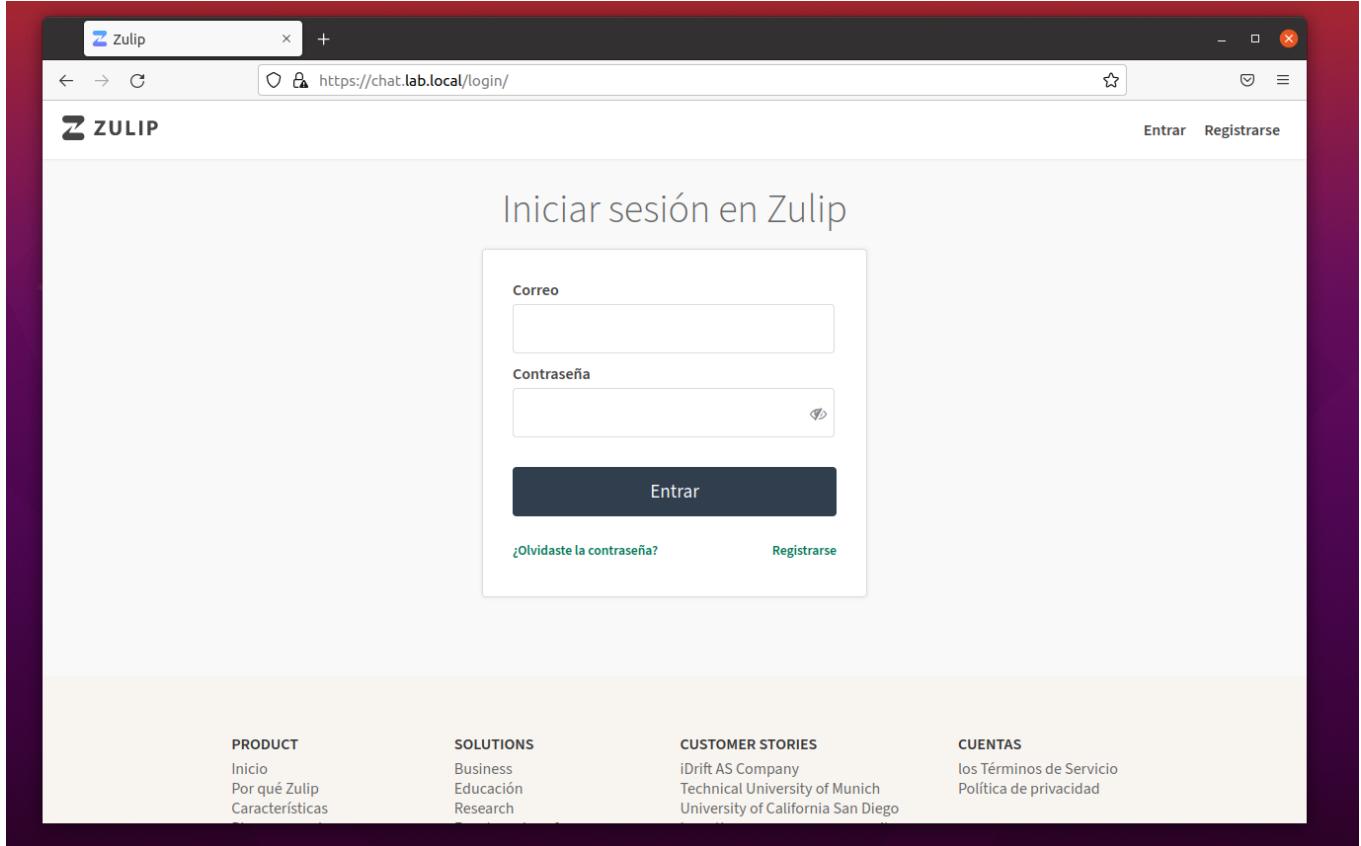
Tras esto, se crea el enlace simbólico para activarlo:

```
ln -s /etc/nginx/sites-available/chat.lab.local /etc/nginx/sites-enabled/chat.lab.local
```

Se comprueba que todo esta correcto y se reinicia el servicio de Nginx y Dnsmasq para aplicar los cambios:

```
nginx -t  
systemctl restart nginx  
systemctl restart dnsmasq
```

Y, desde la máquina **LAPTOP**, visitamos la dirección <https://chat.lab.local> (tras excepcionar el certificado SSL) encontramos:



## Configuración e Instalación de Zulip

Tras comprobar que el Zulip es accesible, se procede a generar un enlace para crear una organización. Esto se debe hacer desde el servidor **VPS**, en la ruta donde hemos creado el archivo docker-compose del zulip:

```
docker compose exec -u zulip zulip /home/zulip/deployments/current/manage.py generate_realm_creation_link
```

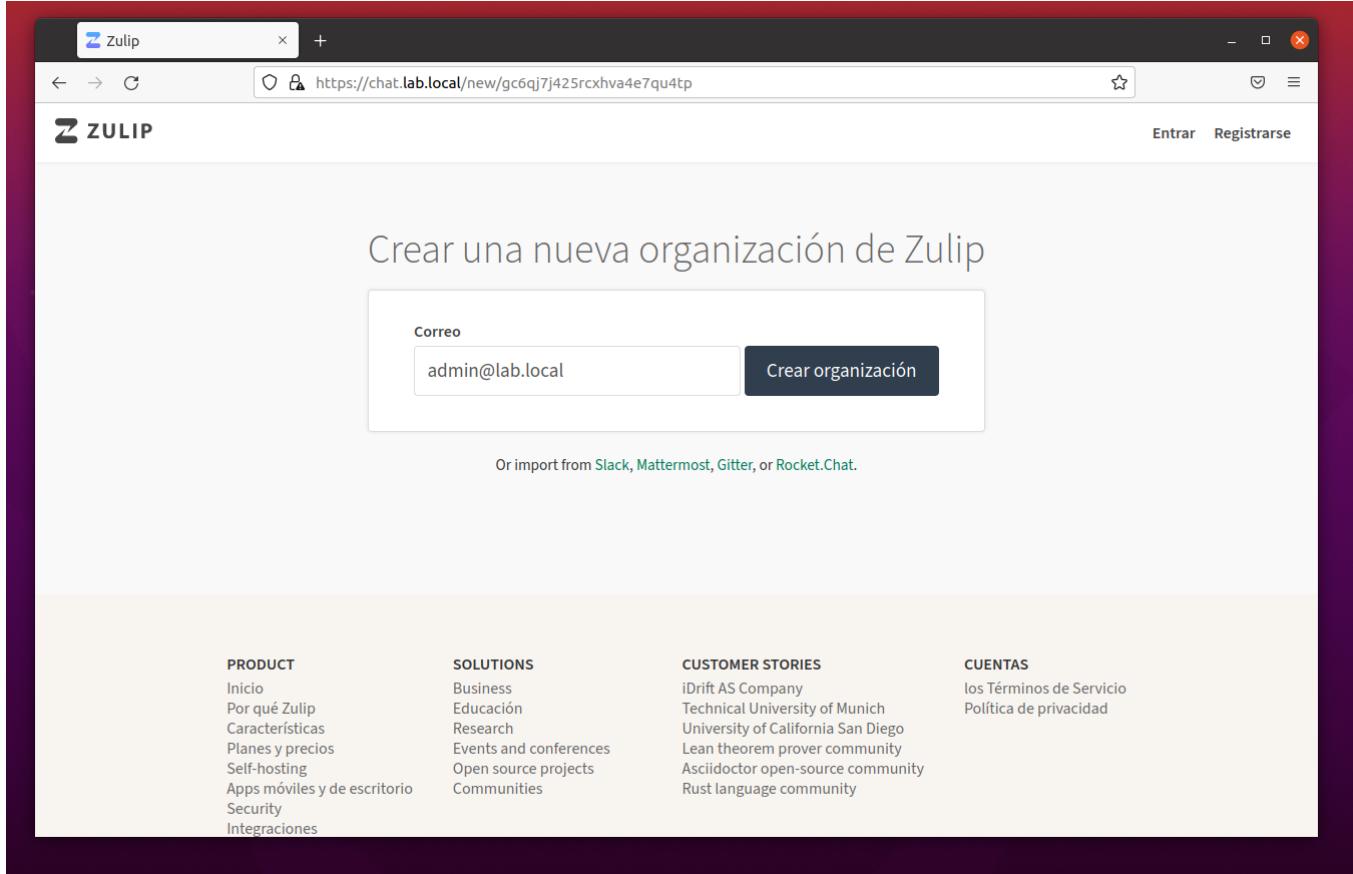
```
root@vps:/srv/containers/zulip# docker compose exec -u zulip zulip /home/zulip/deployments/current/manage.py generate_realm_creation_link
Please visit the following secure single-use link to register your
new Zulip organization:
```

```
https://chat.lab.local/new/gc6qj7j425rcxhva4e7qu4tp
```

```
root@vps:/srv/containers/zulip#
```

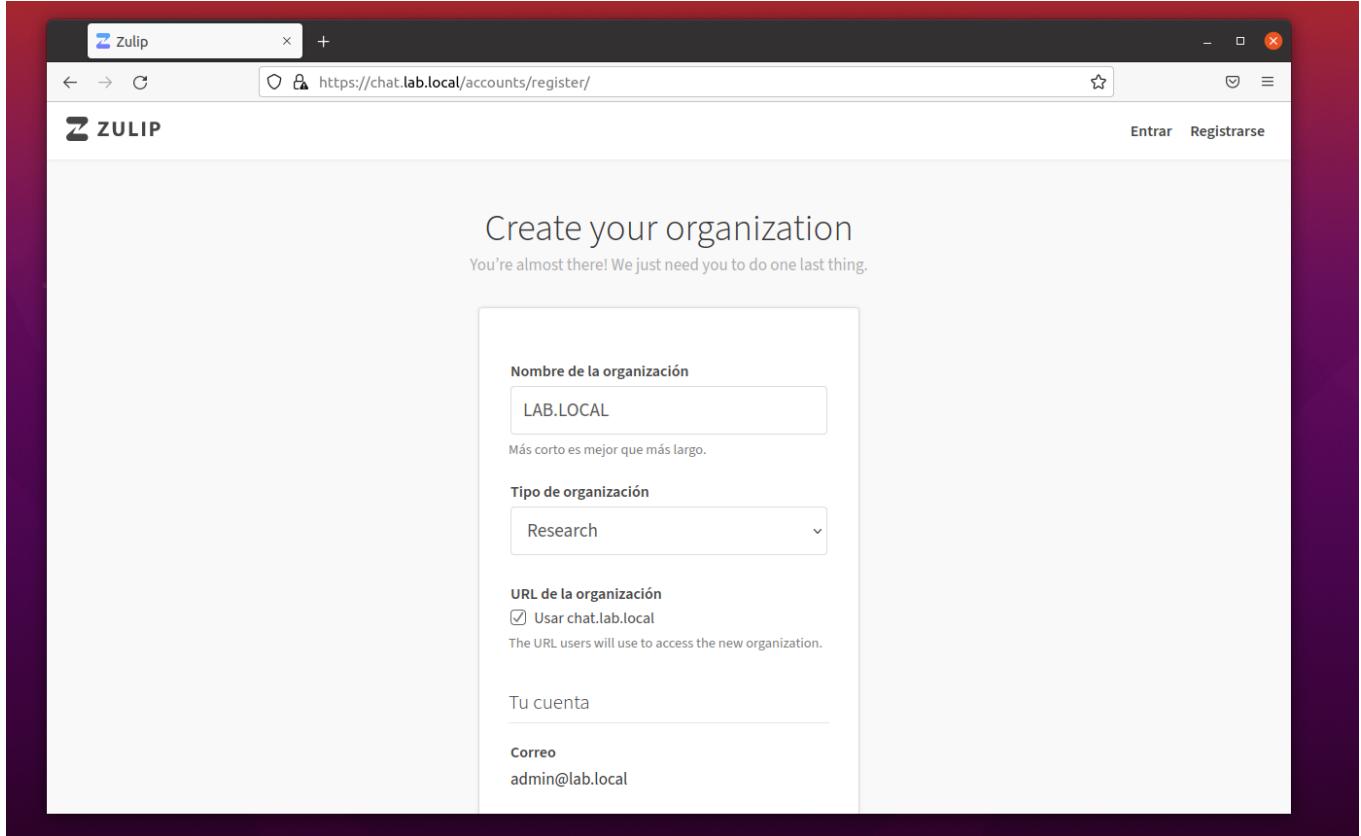
Pegamos este enlace en el navegador de **LAPTOP** e introducimos un correo, en mi caso:

```
admin@lab.local.
```

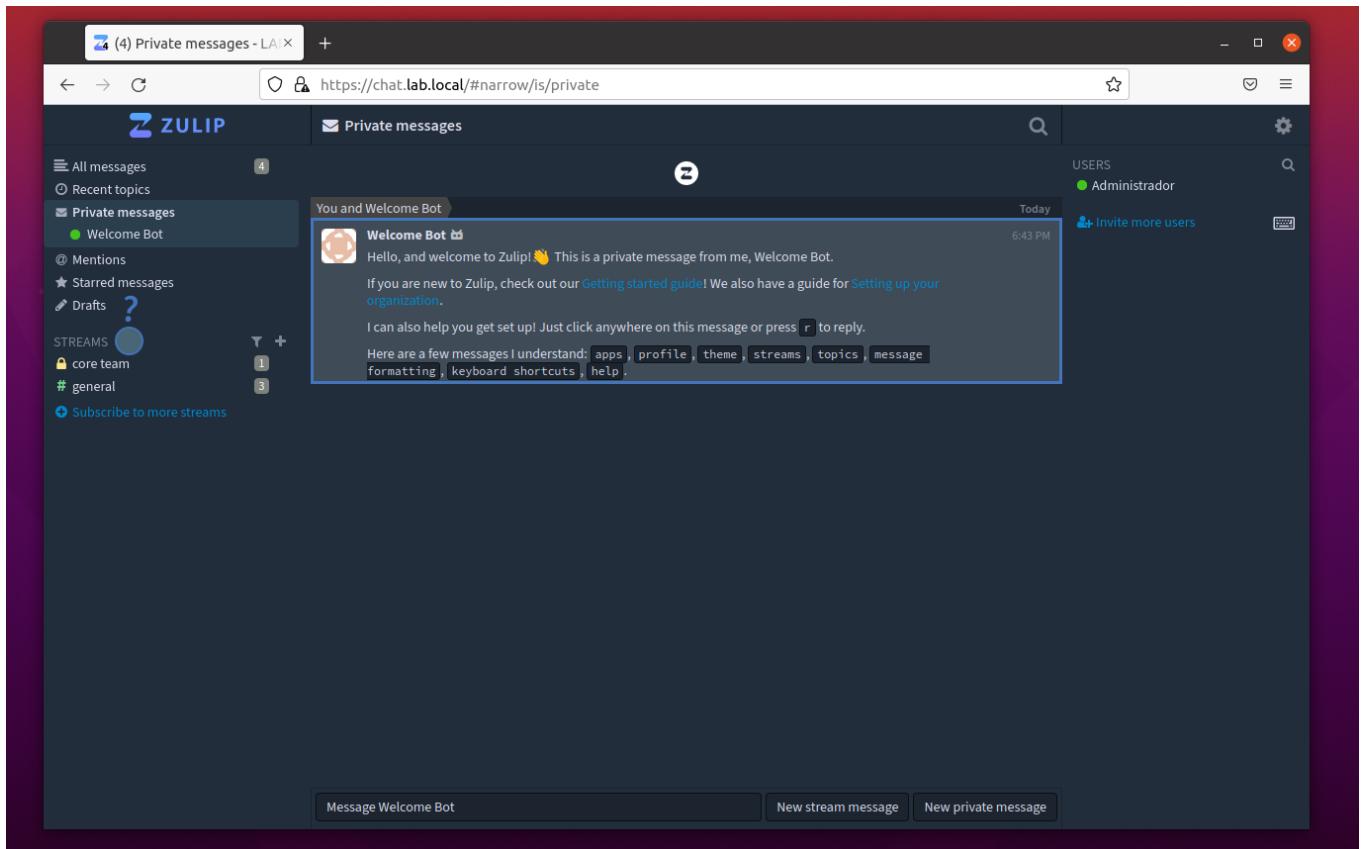


Tras esto, aparecerá la sección para configurar la nueva organización. Aquí ponemos nombre a nuestra organizacion, escogemos el tipo, y una contraseña para el administrador de la organización.

- **Nombre del usuario:** administrador
- **Contraseña:** Adm1nPassw0rd!
- **Correo Electrónico:** [admin@lab.local](mailto:admin@lab.local)



Tras la creación de la cuenta administrativa, se redirige a la vista principal de Zulip como usuario autenticado:



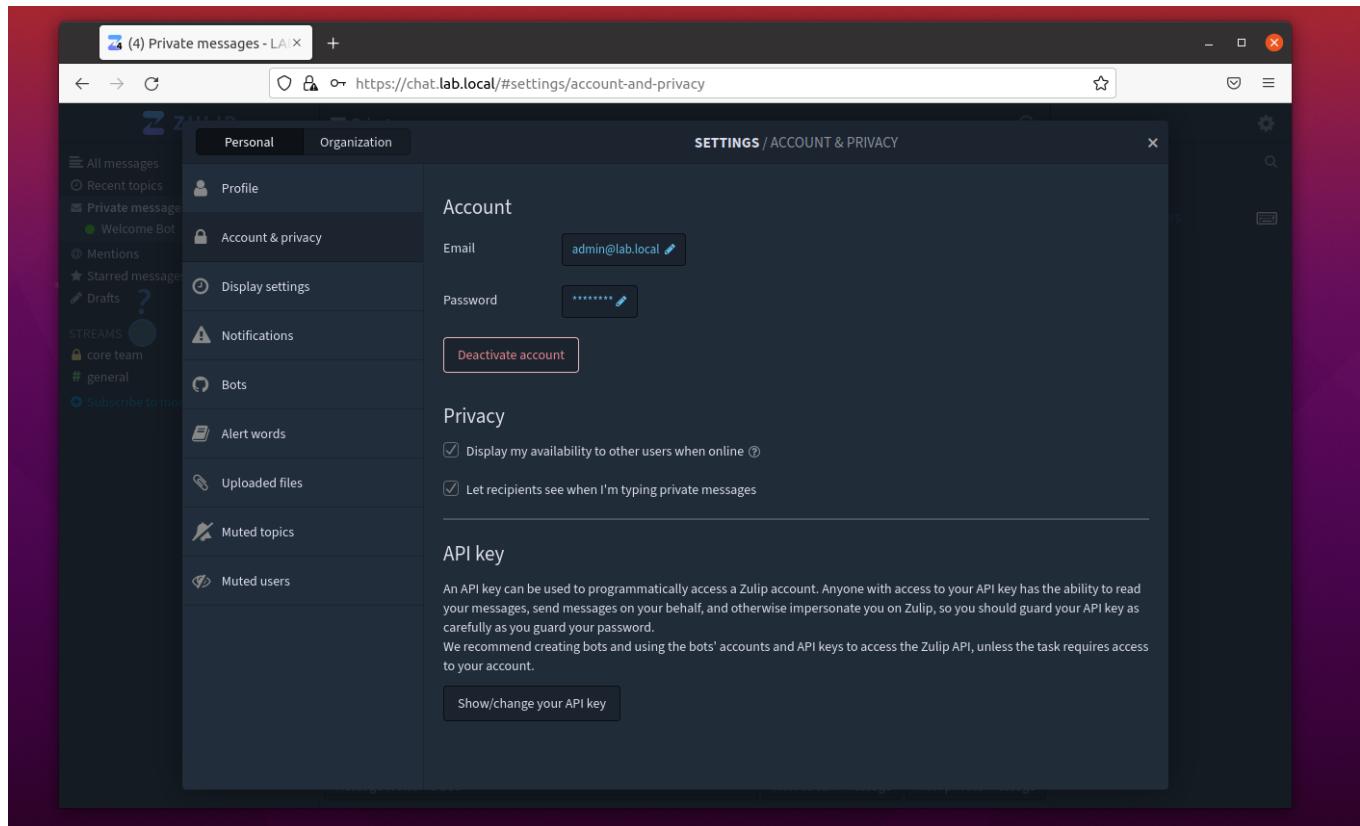
# Organización y gestión de Zulip

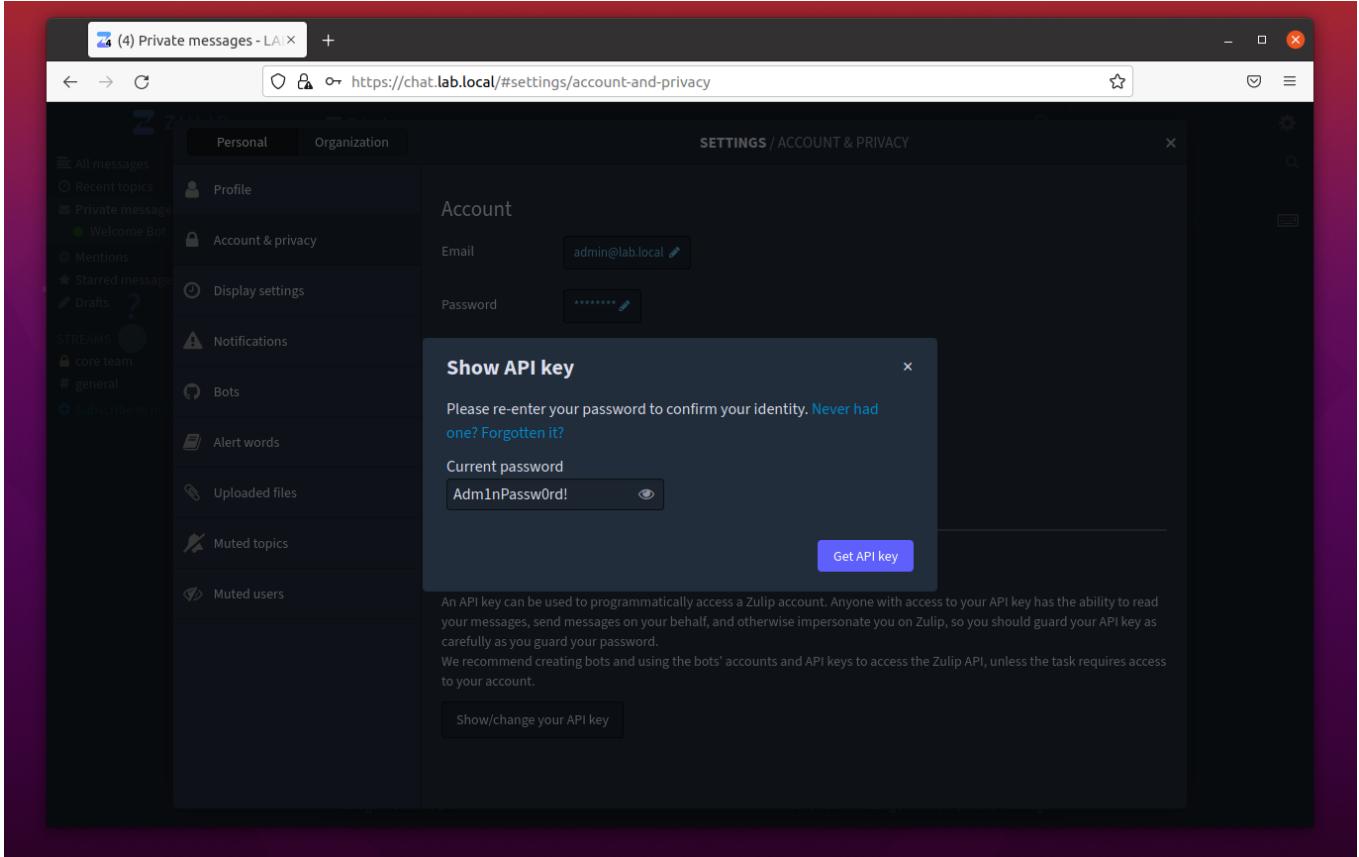
Tras haber concluido con la configuración inicial, se procede a realizar la gestión y organización dentro Zulip.

El primer paso consiste en el **Registro de usuarios dentro de la Organización**. Este proceso es bastante más tedioso de lo que se espera, y es porque Zulip, en el proceso de creación de usuarios o registro, espera una confirmación del usuario a través del correo introducido. Y bien, como en nuestro caso no vamos a levantar ningún servidor SMTP, tendremos que hacer una serie de pasos extra para saltarnos esta política.

Para la creación de usuarios, en lugar de hacerlo a través de la Web UI, utilizaremos la API de Zulip junto con la cuenta del administrador del Zulip.

En primer lugar, procederemos a generar una API Key para el usuario [admin@lab.local](mailto:admin@lab.local) (administrador de nuestro Zulip).





Tras completar el proceso, se obtiene la API KEY = `nPMQzltGoF740ndbCvb8nhJK5MSwEMDi`

### Note

No obstante, el endpoint de la API que permite la creación de usuarios `/api/v1/users`, no permite dar de alta si el usuario autenticado no dispone del rol `can_create_users`. Se puede leer más sobre esto en [la documentación oficial](#)

Por tanto, antes de todo, deberemos asignarnos este permiso. Esta acción sólo puede realizarse a través de la utilidad de administración que se encuentra dentro del contenedor de Zulip. Para realizar la asignación seguimos los siguientes pasos:

En primer lugar, debemos listar los **realm's** gestionados por la instancia Zulip.

### Quote

Los "realms" son el nombre interno de la base de código de Zulip para lo que nos referimos en la documentación de cara al usuario como una organización (el nombre "realm" viene de Kerberos). - cita extraída de [la documentación oficial](#)

Esto se puede hacer con el siguiente comando ejecutando desde la máquina **VPS** en la carpeta del docker-compose de Zulip:

```
docker compose exec -u zulip zulip \  
/home/zulip/deployments/current/manage.py list_realms
```

```
root@vps:/srv/containers/zulip# docker compose exec -u zulip zulip /home/zulip/deployments/current/manage.py list_realms  
id      string_id          name                  domain  
--      -----          ----  
1      zulipinternal        System bot realm      https://zulipinternal.chat.lab.local  
2      ''                   LAB.LOCAL           https://chat.lab.local  
root@vps:/srv/containers/zulip#
```

Una vez localizamos el realm de nuestra organización (**id=2**), procedemos a realizar la asignación del rol a nuestro usuario administrador del Zulip:

```
docker compose exec -u zulip zulip \  
/home/zulip/deployments/current/manage.py change_user_role admin@lab.local  
can_create_users -r 2
```

```
root@vps:/srv/containers/zulip# docker compose exec -u zulip zulip \  
> /home/zulip/deployments/current/manage.py change_user_role admin@lab.local can_create_users -r 2  
root@vps:/srv/containers/zulip#
```

Tras realizar la asignación, procedemos a entrar dentro del contenedor de Zulip para realizar la petición HTTP contra la API que nos permitirá crear el nuevo usuario. Entramos al contenedor con:

```
docker compose exec -u zulip zulip bash
```

Después realizamos la petición con cURL desde el contenedor con el siguiente comando:

### Note

Recuerda poner la API KEY obtenida y los datos del usuario nuevo como se ve a continuación.

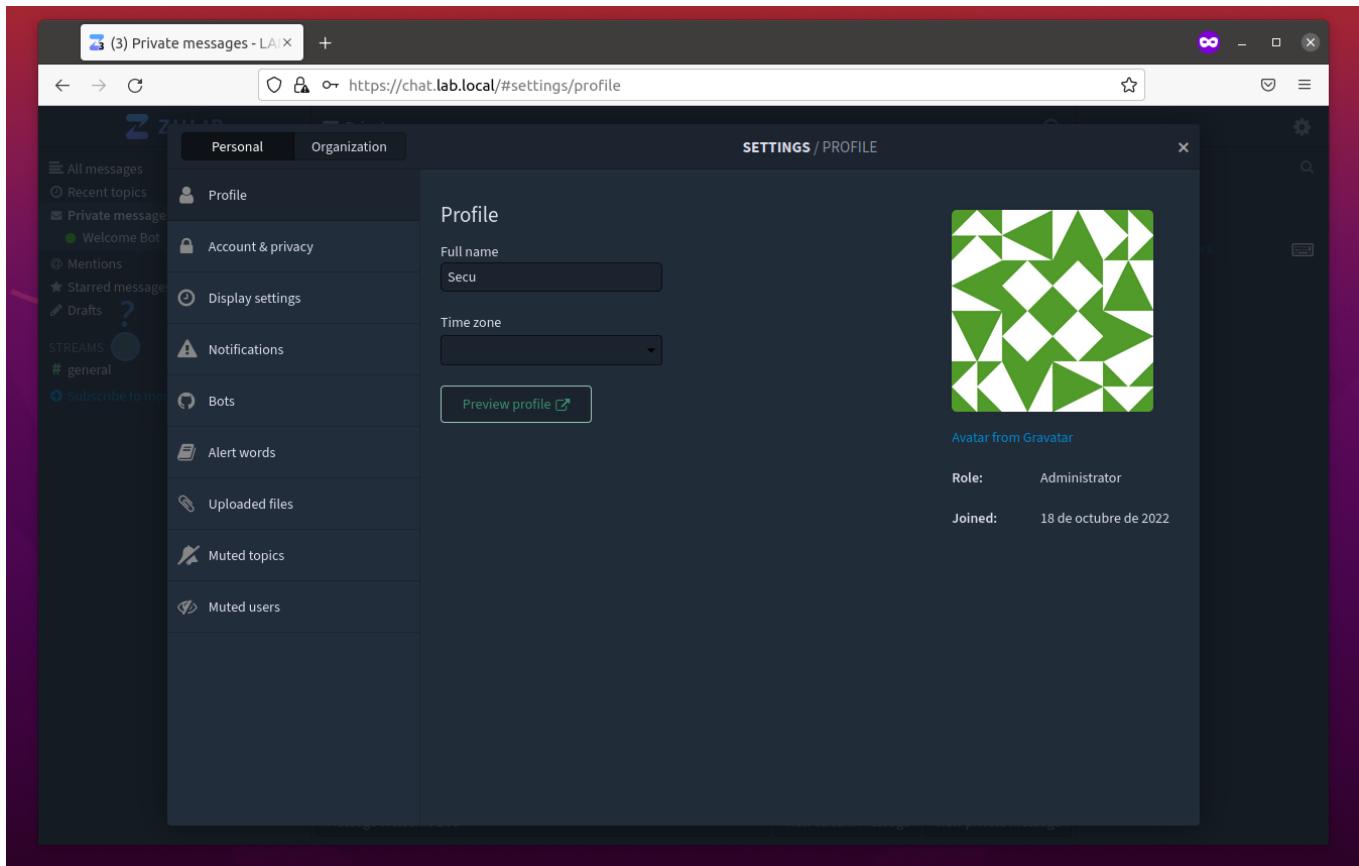
```
curl -ksSX POST https://127.0.0.1/api/v1/users \  
-u admin@lab.local:nPMQzltGoF740ndbCvb8nhJK5MSwEMDi \  
--data-urlencode 'email=secu@lab.local' \  
--data-urlencode 'password=Passw0rd00' \  
--data-urlencode 'full_name=Secu'
```

```
root@vps:/srv/containers/zulip# docker compose exec -u zulip zulip bash
zulip@5548df454f69:~$ curl -ksSX POST https://127.0.0.1/api/v1/users \
> -u admin@lab.local:nPMQzltGoF740ndbCvb8nhJK5MSwEMDi \
> --data-urlencode 'email=secu@lab.local' \
> --data-urlencode 'password=Passw0rd00' \
> --data-urlencode 'full_name=Secu'
{"result":"success","msg":"","user_id":9}
zulip@5548df454f69:~$
```

Si volvemos a la máquina **LAPTOP** y vamos a la sección de **Usuarios de la Organización** dentro de la configuración con el usuario administrador, podemos ver el nuevo usuario creado:

| Name                | Email           | User ID | Role   | Last active | Actions |
|---------------------|-----------------|---------|--------|-------------|---------|
| Administrador (you) | admin@lab.local | 8       | Owner  | Today       |         |
| Secu                | secu@lab.local  | 9       | Member | Unknown     |         |

Finalmente iniciamos sesión con credenciales utilizadas en la creación del usuario y accedemos como el usuario **secu**:



### Note

El usuario nuevo simplemente deberá cambiar la contraseña y ya estaría. En caso de querer eliminar un usuario, se podría hacer de la misma forma con el comando:

```
docker compose exec -u zulip zulip
/home/zulip/deployments/current/manage.py manage.py delete_user -u
secu@lab.local -f
```

Con los usuarios creados, se procede a explicar una de las características clave de Zulip, que son: los **topics** y **streams**.

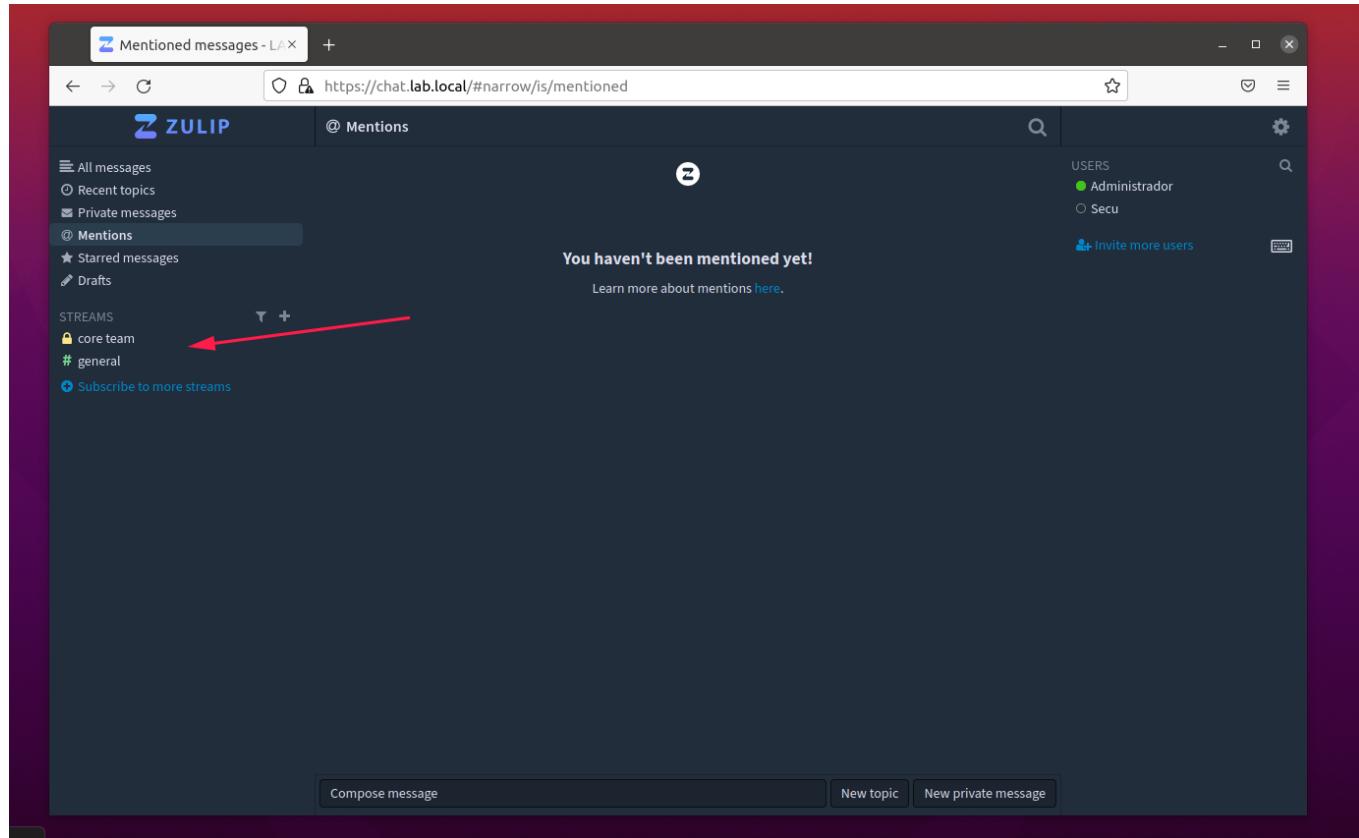
- Los **Streams** en Zulip son algo parecido a lo que podrían ser los "canales". Dentro de un Stream pueden encontrarse los **Topics**.
- Los **Topics** en Zulip son como los "hilos", es decir, conversaciones dentro de un Stream para hablar sobre un tema o aspecto.

Los Topics y Streams permiten organizar la comunicación dentro de una organización y separar los temas en sus categorías correspondientes. Incluso dentro de estas categorías, se pueden tratar diferentes temas (topics).

Un ejemplo de uso de estas funcionalidades podría ser el siguiente:

- Dentro de una organización, se utilizan los Streams como "canales de equipo". Cada equipo tiene un Stream, por ejemplo: existe un stream "blue team", "red team", "pentest", "it", "devel", etc.
- En un Canal de equipo, por ejemplo, "red team", se organizan los distintos temas en "topics", por ejemplo: "evasion", "infra", "opsec", "develop", "osint", etc.

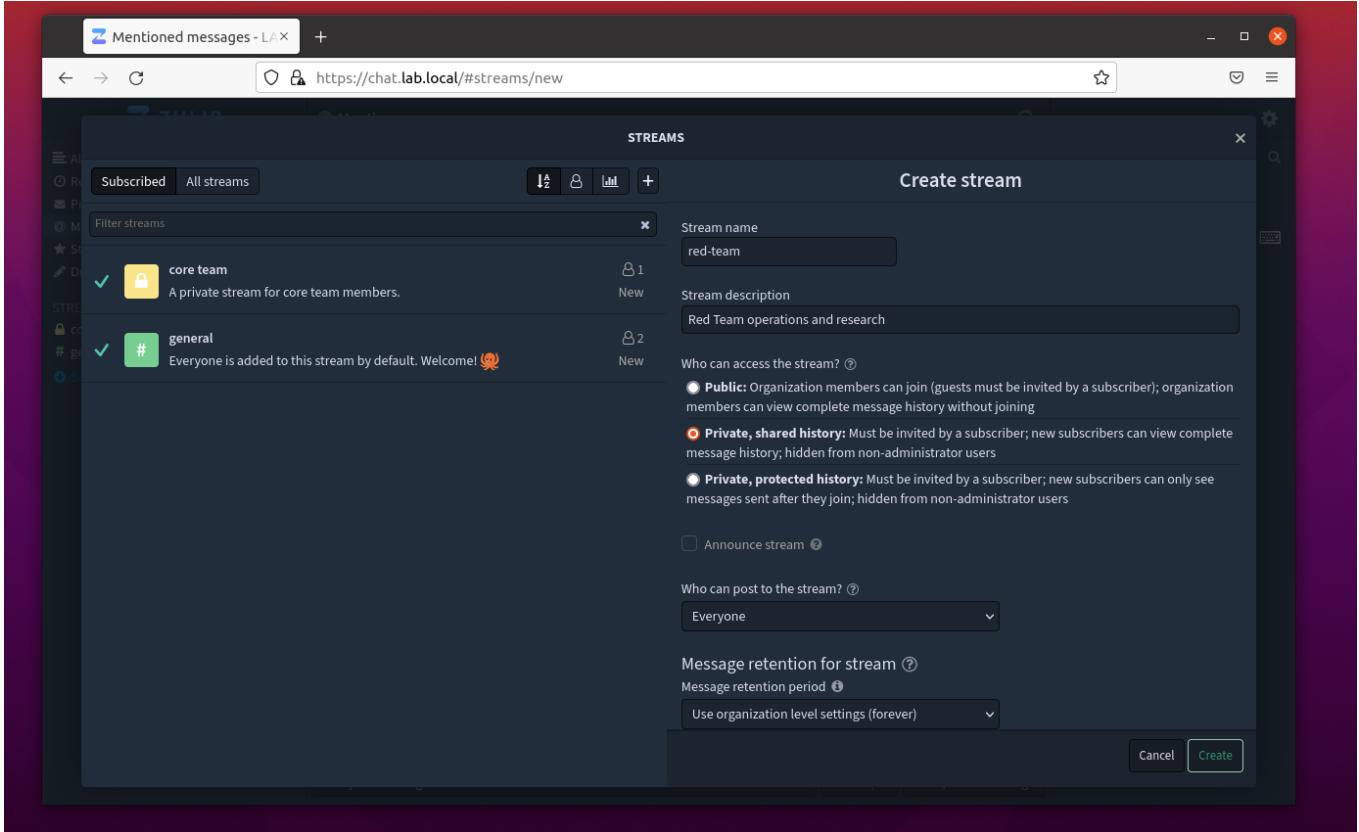
Los Streams se pueden encontrar en el lateral izquierdo dentro de la vista de Zulip:



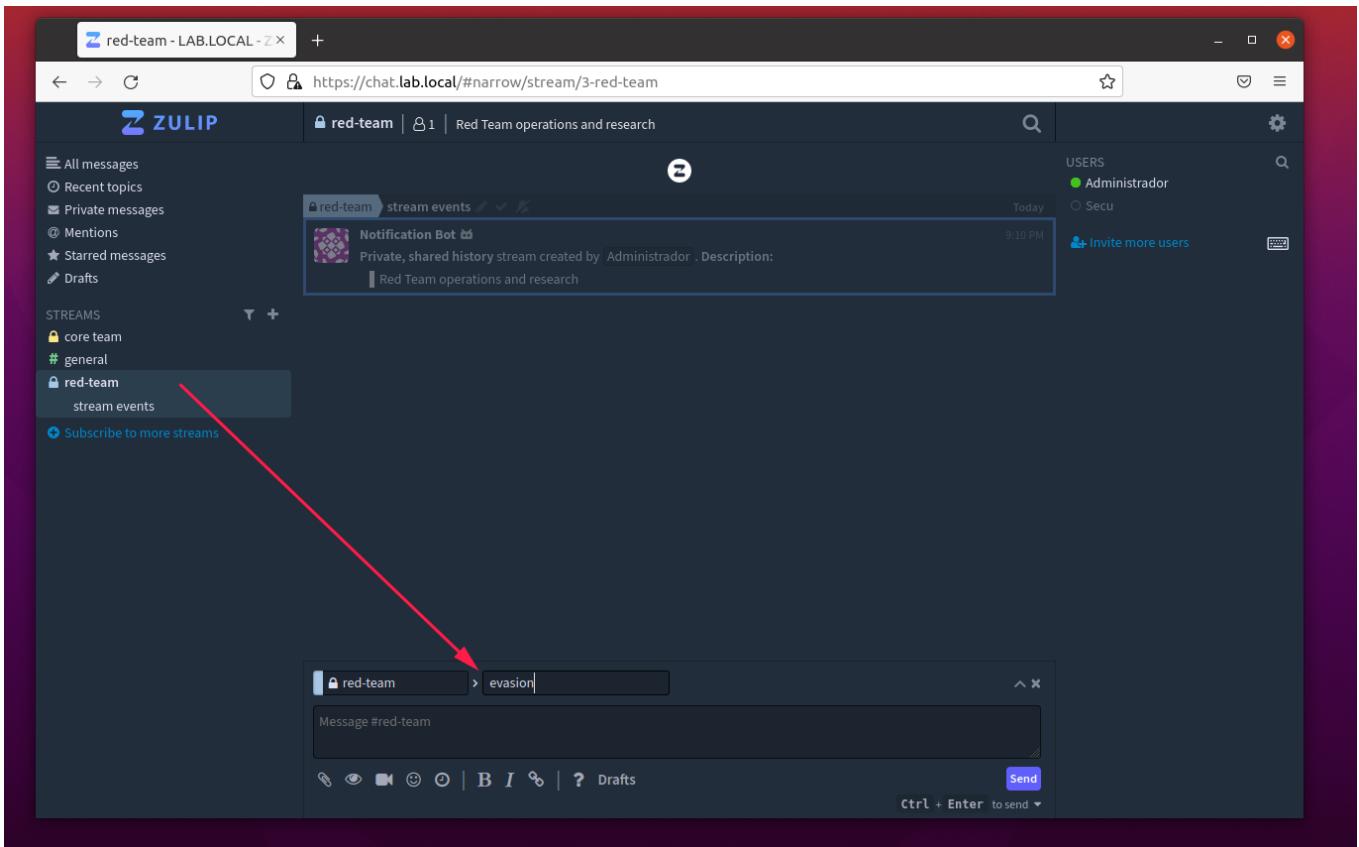
### Note

Los **streams** se identifican en la organización por iconos y colores. Al crear un stream esta asignación parece ser aleatoria (en el momento de la guía yo no he conseguido cambiar los colores ni los iconos, pero intuyo que es posible hacerlo).

La creación del stream es bastante sencilla y los campos se explican por si solos:



Tras crear el Stream, se puede proceder a desplegarlo para crear un topic. En este caso yo crearé el topic **evasion** dentro del stream **red team**:



Puedes ver los topics en conjunto haciendo click en el stream, o también ver sólo un topic haciendo click sobre él:

The image contains two screenshots of the Zulip messaging platform interface, demonstrating how to view topics.

**Screenshot 1 (Top):** Shows the "red-team" stream selected in the sidebar. A red arrow points to the "infra" topic within the stream. The message list shows a message from "Administrador" with the subject "Detalles de la infraestructura del Red Team".

| User          | Topic | Message Content                             | Time    |
|---------------|-------|---|---------|
| Administrador | infra | Detalles de la infraestructura del Red Team | 9:17 PM |

**Screenshot 2 (Bottom):** Shows the "red-team" stream selected in the sidebar. A red arrow points to the "red-team" stream header. The message list shows three messages: one from "Administrador" in the "evasion" topic, one from "Administrador" in the "osint" topic, and one from "Administrador" in the "infra" topic.

| User          | Topic   | Message Content                             | Time    |
|---------------|---------|---|---------|
| Administrador | evasion | Stream para cosas de Evasión                | 9:16 PM |
| Administrador | osint   | Aquí cosas de OSINT                         | 9:17 PM |
| Administrador | infra   | Detalles de la infraestructura del Red Team | 9:17 PM |

Finalmente, una apreciación a comentar es el tema de la API de Zulip (que ya hemos utilizado anteriormente). El uso de esta API permite definir acciones automáticas a realizar, lo que mejora la funcionalidad del servicio al poder integrarse con otras herramientas.

## Servicio de Gestión de Credenciales - Syspass



# ¿Por qué utilizar Syspass?

## jj Quote

"Es una potente aplicación web que proporciona una gestión de contraseñas de forma segura y colaborativa. Cuenta con muchas opciones para facilitar y reforzar la seguridad a la hora de compartir contraseñas entre equipos, departamentos o clientes, incluyendo ACLs, perfiles, campos personalizados, valores preestablecidos o enlaces públicos, entre otros muchos útiles." - extracto de [la página oficial de Syspass](#)

Sysspass es una solución que nos permitirá realizar una gestión de claves dentro de una organización de forma eficiente y controlada. Esta desarrollada en PHP y utiliza HTML5 como Frontend; tiene un diseño muy intuitivo y disponen de una versión alojada en la nube que puedes probar: [demo.sysspass.org](http://demo.sysspass.org)

Los motivos por los cuales he decidido utilizar sysspass frente a otros servicios de gestión de credenciales son:

1. La sencillez y facilidad de despliegue on-premises de la aplicación web (pues consta de un docker-compose bastante simple y puedes encontrar muchas documentación acerca del despliegue en su [wiki oficial](#)).
2. La gestión que permite realizar a partir de los **grupos, perfiles, acls, etc** permite organizar una infraestructura de forma eficiente y sencilla. El control que aporta la herramienta es el motivo principal por el cuál creo que destaca sobre otras.
3. Dispone de otras funcionalidades que, aunque no haya usado, me parecen de interés pues permite adaptar la aplicación a diferentes tipos de organizaciones. Estas son: soporte con autenticación LDAP (OpenLDAP, eDirectory, Active Directory, freeIPA, etc), permite configurar un segundo factor de autenticación (Google Autenticator) y dispone de una API para diferentes gestiones de usuarios, grupos, etc.
4. También incorpora una [estrategia de backups](#) que especifica cómo deben realizarse los backups en un despliegue on-premises.

Finalmente, recomiendo echar un vistazo a su [documentación oficial](#), dónde puede encontrarse todo el detalle acerca de las capacidades que aporta esta herramienta.

## Despliegue de contenedor de Sysspass

Comenzamos el despliegue del contenedor de Sysspass utilizando el siguiente archivo docker-compose.yml que se ve a continuación:

```
version: '3'
services:
  app:
    container_name: syspass-app
    image: syspass/syspass:3.2.11
    restart: always
    ports:
      - "127.0.0.1:8444:443"
    depends_on:
      - db
    volumes:
      - /srv/data/syspass/syspass-config:/var/www/html/sysPass/app/config
      - /srv/data/syspass/syspass-backup:/var/www/html/sysPass/app/backup
    environment:
      - USE_SSL=yes
      - COMPOSER_EXTENSIONS=syspass/plugin-authenticator
  db:
    container_name: syspass-db
    restart: always
    image: mariadb:10.2
    environment:
      - MYSQL_ROOT_PASSWORD=MySQLP4ssw0rd
    expose:
      - "3306"
    volumes:
      - /srv/data/syspass/syspass-db:/var/lib/mysql
```

### Note

Recuerda cambiar la contraseña de la base de datos por una lo suficientemente segura. La que aquí figura es a modo de ejemplo para la guía.

Este archivo YAML, lo vamos a situar en la ruta: `/srv/containers/syspass/docker-compose.yml`. Antes de desplegar el contendor, creamos la ruta del volumen en el directorio de datos:

```
mkdir -p /srv/data/syspass/
```

Para proceder a su despliegue simplemente ejecutamos el siguiente comando:

```
docker compose up -d
```

Tras el despliegue podemos ver como el contenedor esta corriendo con:

```
docker compose ps
```

| NAME        | COMMAND                             | SERVICE | STATUS  | PORTS                           |
|-------------|-------------------------------------|---------|---------|---------------------------------|
| syspass-app | "/usr/local/sbin/entrypoint.sh app" | app     | running | 80/tcp, 127.0.0.1:8444->443/tcp |
| syspass-db  | "docker-entrypoint.s..."            | db      | running | 3306/tcp                        |

Se añade el subdominio al `/etc/hosts` para gestionar el servicio (como se ha hecho anteriormente) quedando el archivo así:

```
127.0.0.1 localhost
127.0.1.1 vps

# The following lines are desirable for IPv6 capable hosts
::1      ip6-localhost ip6-loopback
fe00::0  ip6-localnet
ff00::0  ip6-mcastprefix
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters

# LAB.LOCAL DNS records
192.168.10.1 vps.lab.local
192.168.10.2 laptop.lab.local

192.168.10.1 git.lab.local
192.168.10.1 md.lab.local
192.168.10.1 chat.lab.local
192.168.10.1 pass.lab.local
```

También se crea el archivo de configuración de Nginx para gestionar el virtual host. Este archivo se sitúa en `/etc/nginx/sites-available/pass.lab.local` y tiene el siguiente contenido:

```
server {
    listen 192.168.10.1:80;
    server_name pass.lab.local;
    # Redirect all non-https requests
    rewrite ^ https://$host$request_uri? permanent;
}

server {
    listen 192.168.10.1:443 ssl http2;

    include snippets/self-signed.conf;
    include snippets/ssl-params.conf;
```

```
server_name pass.lab.local;

client_max_body_size 0;
proxy_http_version 1.1;
proxy_request_buffering off;

location / {
    proxy_pass https://127.0.0.1:8444;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}
}
```

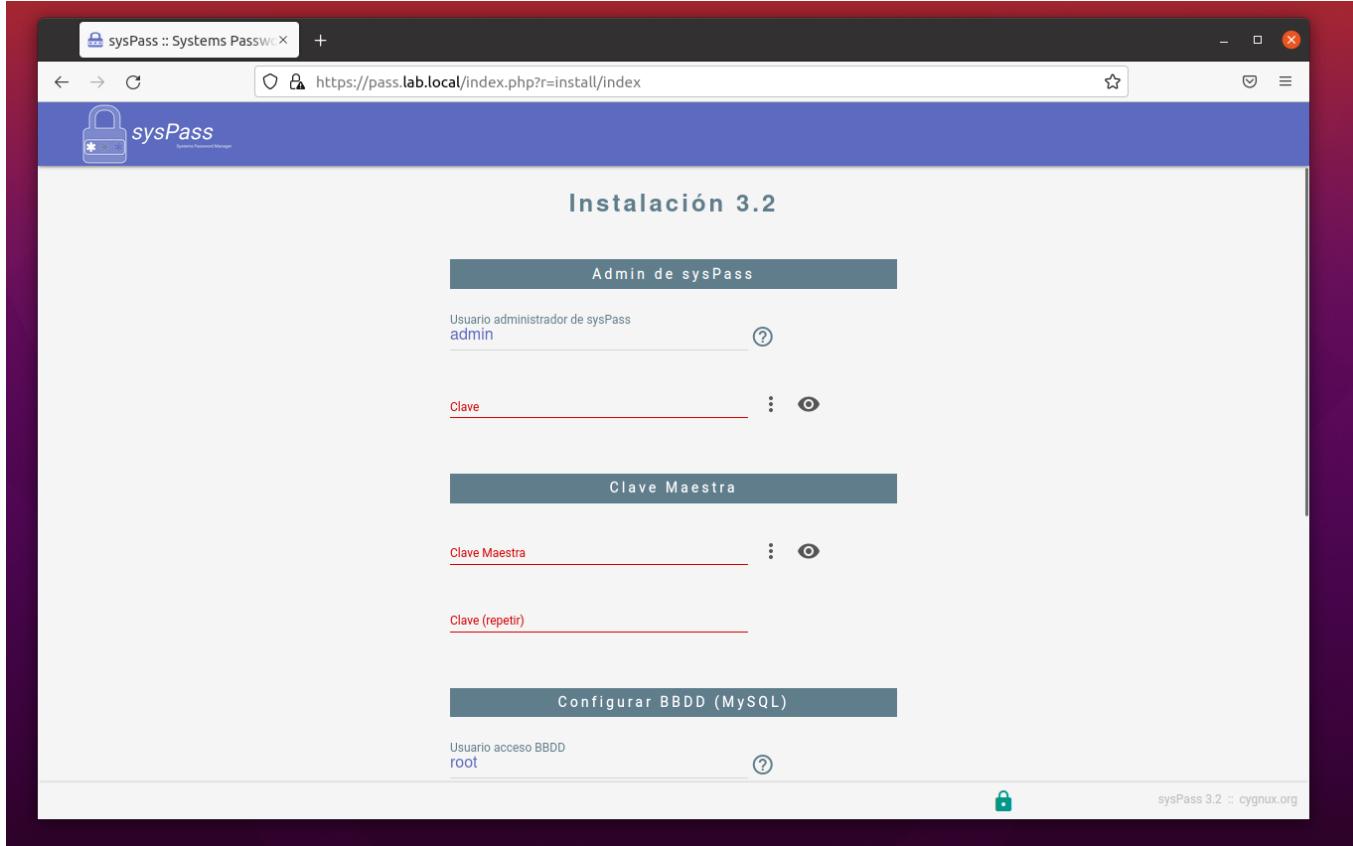
Tras esto, se crea el enlace simbólico para activarlo:

```
ln -s /etc/nginx/sites-available/pass.lab.local /etc/nginx/sites-enabled/pass.lab.local
```

Se comprueba que todo esta correcto y se reinicia el servicio de Nginx y Dnsmasq para aplicar los cambios:

```
nginx -t
systemctl restart nginx
systemctl restart dnsmasq
```

Y, desde la máquina **LAPTOP**, visitamos la dirección <https://pass.lab.local> (tras excepcionar el certificado SSL) encontramos:

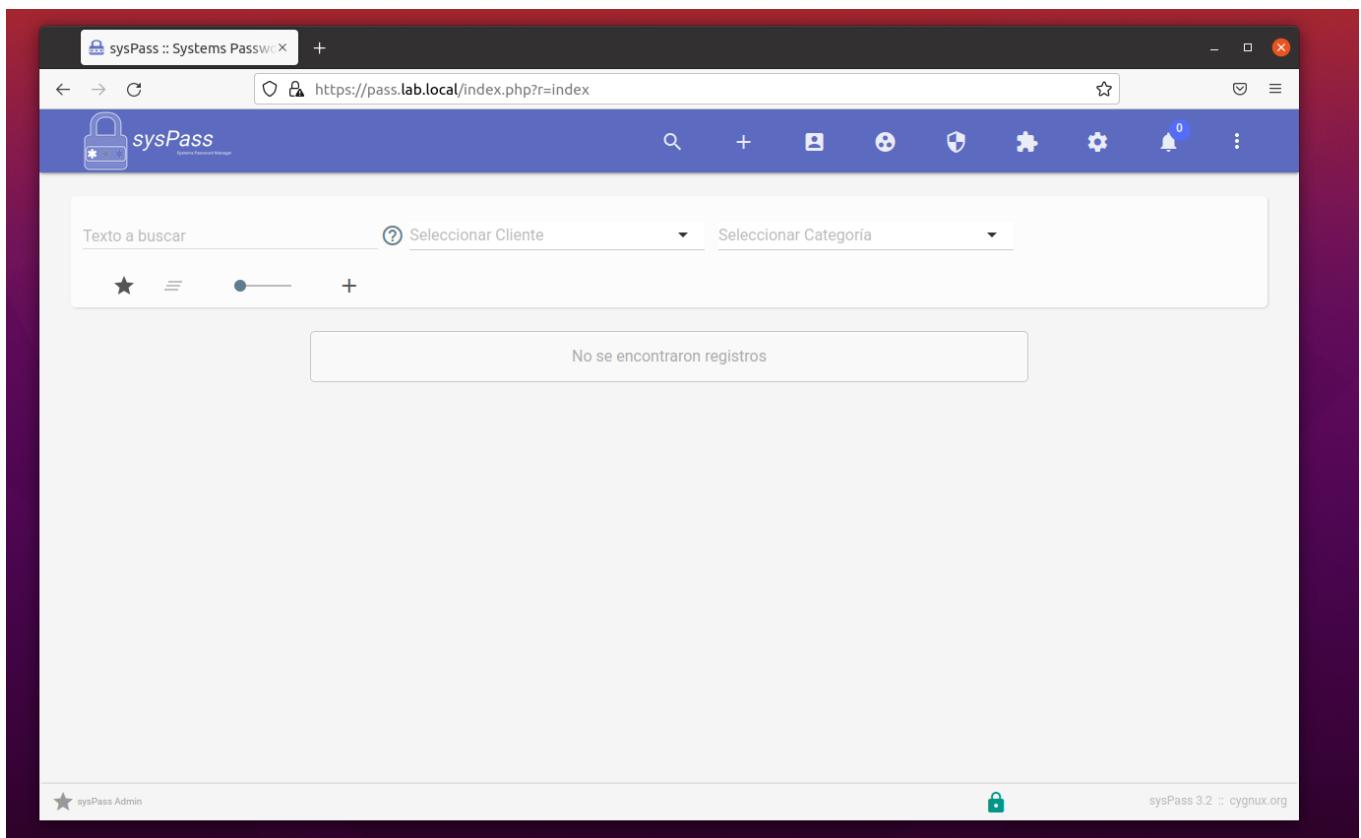
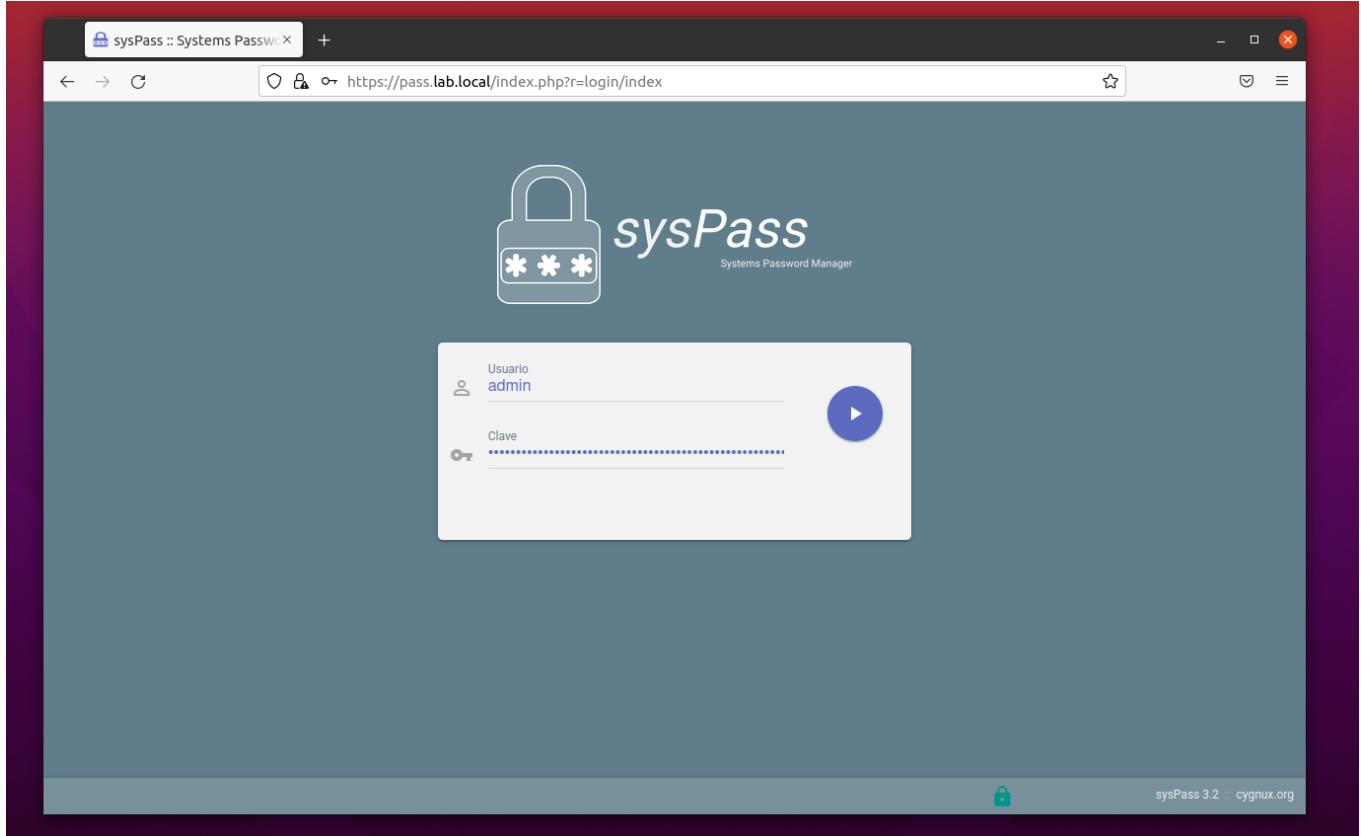


## Configuración e instalación de Syspass

En la sección principal, se nos pide la información inicial para configurar Syspass. A lo que nosotros responderemos con unos datos como los siguientes:

- Usuario administrador: `admin`
- Clave del administrador: `Adm1nP4ssw0rd!`
- Clave Maestra: `M4st3rS3ccur3K3y!`
- Servidor DDBB para Syspass: `db` (este es importante que sea exactamente así)

Tras completar la configuración con los datos correspondientes, se redirigirá al panel de login, donde introduciremos las credenciales del administrador:



Hasta aquí llega el proceso de configuración e instalación de Syspass.

# Gestión y organización de Syspass

Para exemplificar la estructuración de credenciales, grupos y usuarios dentro de Syspass procederemos a recrear la siguiente estructura:

- Grupo **Red Team**
  - Miembros:
    - Secury (Red Team Manager)
    - Vici (Red Team Operator)
    - Gygas (Red Team Operator)
  - Claves:
    - RT OSINT
      - Censys Search
      - Shodan
      - Security Trails
    - RT Infraestructure
      - Gophish
      - Cobaltstrike Teamserver
- Grupo **Infraestructura**
  - Miembros:
    - L (Sysadmin Manager)
    - Jack (Sysadmin Assistant)
  - Claves:
    - Sys Infrastructure
      - Syspass Admin
      - Syspass Mysql Root

## Note

Esta estructura (aunque mínima) podría considerarse parte de la organización que se podría definir dentro de una empresa u organización. Separando entre grupos de distintos departamentos, y otorgando acceso a determinados usuarios a conjuntos de claves.

Antes de comenzar, se destacan algunas de las funcionalidades principales de la estructura anterior:

- Los **Perfiles** nos permitirán definir el rol de cada usuario creado en Syspass (por ejemplo: Red Team Manager, Red Team Operator, Sysadmin Manager, etc).
- Los **Grupos** definirán la separación y organización de los distintos departamentos del equipo (por ejemplo: Red Team, Infraestructura, etc).
- Los **Cientes**, aunque en este caso no se utilizan más que superficialmente, puede suponer tanto una forma de organización interna como de clientes de la propia empresa u organización (en este ejemplo se consta de un único cliente, y es LAB.LOCAL).
- Finalmente, las **Categorías** permiten agrupar conjuntos de claves que comparten un cierto valor entre si (por ejemplo: OSINT, RT Infraestructura, etc).

El orden de los pasos que seguiremos para la definición de este escenario, será el siguiente:

1. Creación de Perfiles
2. Creación de Grupos
3. Creación de Usuarios
4. Creación de Clientes
5. Creación de Categorías
6. Creación de Claves

Para no extender demasiado el proceso, simplemente se mostrará el resultado final del como quedaría la configuración de los puntos anteriores.

Tras la creación de perfiles, debemos obtener algo como lo siguiente:

The screenshot shows the sysPass web interface with the 'PERFILES' tab selected. The main content area displays a list of user profiles:

| Nombre             |
|--------------------|
| Admin              |
| Red Team Manager   |
| Red Team Operator  |
| Sysadmin Assistant |
| Sysadmin Manager   |

At the bottom right of the interface, there is a footer bar with the text "sysPass Admin" and "sysPass 3.2 :: cygnux.org".

Luego, los grupos quedan definidos como:

The screenshot shows the sysPass web interface with the 'GRUPOS' tab selected. The main content area displays a list of groups:

| Nombre          | Descripción           |
|-----------------|-----------------------|
| Admins          | sysPass Admins        |
| Infraestructura | Infrastructure Admins |
| Red Team        | Red Team Operators    |

At the bottom right of the interface, there is a footer bar with the text "sysPass Admin" and "sysPass 3.2 :: cygnux.org".

Siguiendo con la creación de usuarios, el resultado tiene la siguiente forma:

| Nombre        | Login | Perfil             | Grupo           | Propiedades |
|---------------|-------|--------------------|-----------------|-------------|
| Gygas         | gygas | Red Team Operator  | Red Team        |             |
| Jack          | jack  | Sysadmin Assistant | Infraestructura |             |
| L             | l     | Sysadmin Manager   | Infraestructura |             |
| Security      | secu  | Red Team Manager   | Red Team        |             |
| sysPass Admin | admin | Admin              | Admins          |             |
| Vici          | vici  | Red Team Operator  | Red Team        |             |

La creación de clientes, en este caso se limita a un sólo valor (nuestra propia organización):

| Nombre    | Descripción        | Global |
|-----------|--------------------|--------|
| LAB.LOCAL | Internal LAB.LOCAL | SI     |

Después, plasmamos las siguientes categorias:

sysPass :: Systems Password Manager

https://pass.lab.local/index.php?r=index

CATEGORÍAS ETIQUETAS CLIENTES CAMPOS PERSONALIZADOS ARCHIVOS CUENTAS CUENTAS (H) VALORES PREDETERMINADOS

| Nombre             | Descripción              |  |
|--------------------|--------------------------|--|
| RT Infrastructure  | Red Team Infraestructure |  |
| RT OSINT           | Red Team OSINT resources |  |
| Sys Infrastructure | Sysadmin Infraestructure |  |

3 @ 0,00145s 1 / 1 (12)

sysPass Admin sysPass 3.2 :: cygnux.org

Y finalmente, se proceden a crear las claves que hemos definido:

sysPass :: Systems Password Manager

https://pass.lab.local/index.php?r=index

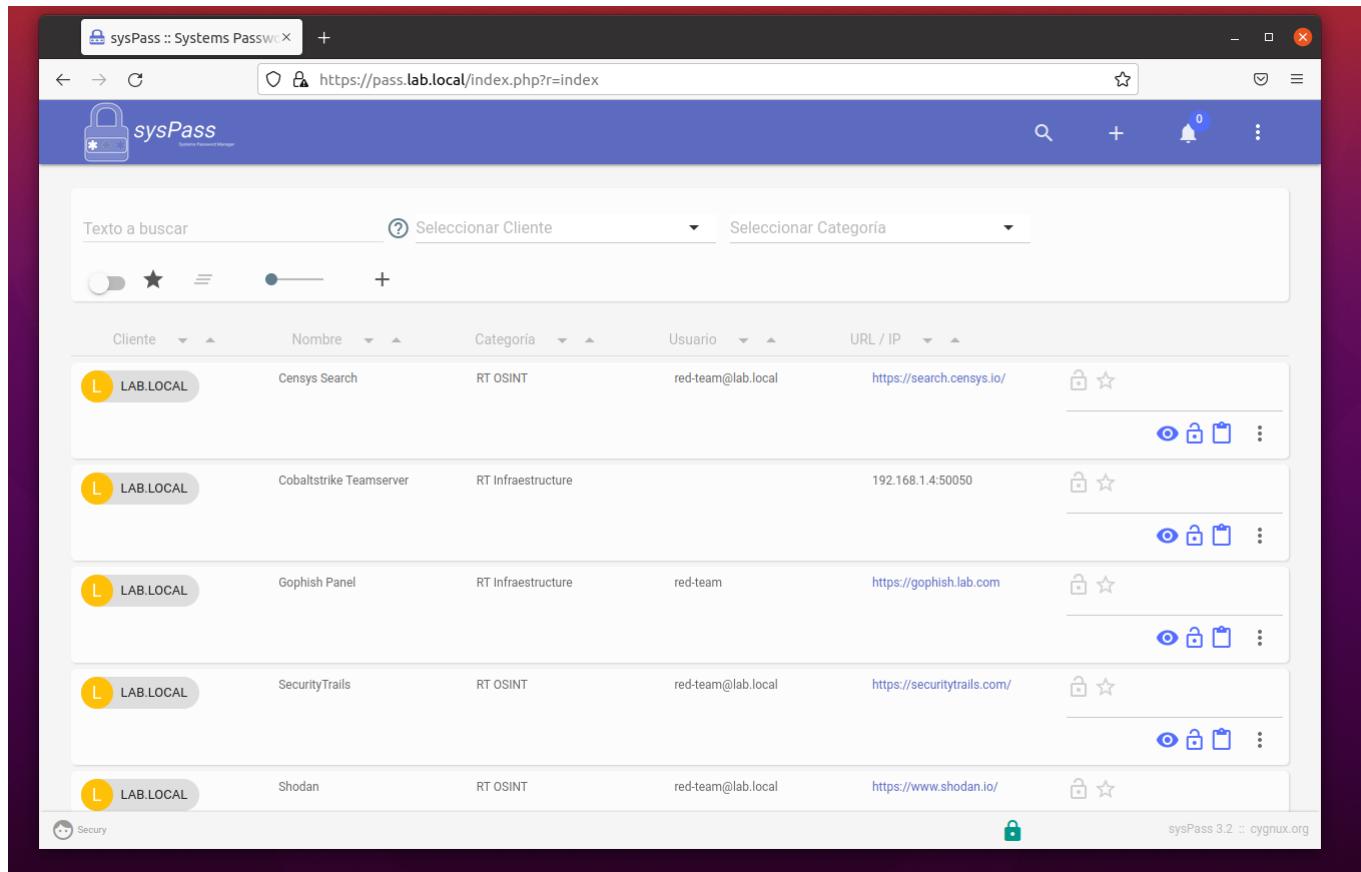
CATEGORÍAS ETIQUETAS CLIENTES CAMPOS PERSONALIZADOS ARCHIVOS CUENTAS CUENTAS (H) VALORES PREDETERMINADOS

| Nombre                  | Cliente   | Categoría           | Propietario   | Grupo Principal |  |
|-------------------------|-----------|---------------------|---------------|-----------------|--|
| Censys Search           | LAB.LOCAL | RT OSINT            | sysPass Admin | Red Team        |  |
| Cobaltstrike Teamserver | LAB.LOCAL | RT Infraestructure  | sysPass Admin | Red Team        |  |
| Gophish Panel           | LAB.LOCAL | RT Infraestructure  | sysPass Admin | Red Team        |  |
| SecurityTrails          | LAB.LOCAL | RT OSINT            | sysPass Admin | Red Team        |  |
| Shodan                  | LAB.LOCAL | RT OSINT            | sysPass Admin | Red Team        |  |
| Syspass Admin           | LAB.LOCAL | Sys Infraestructure | sysPass Admin | Infraestructura |  |
| Syspass Database        | LAB.LOCAL | Sys Infraestructure | sysPass Admin | Infraestructura |  |

7 @ 0,00157s 1 / 1 (12)

sysPass Admin sysPass 3.2 :: cygnux.org

Después de completar la configuración y creación de claves; al acceder con un usuario, como por ejemplo Secury, puede ver las claves creadas (a las que tiene acceso) en la sección principal.



The screenshot shows the sysPass web application interface. At the top, there is a header bar with the title "sysPass :: Systems Password Manager". Below the header, there is a search bar and a notification bell icon. The main content area displays a table of generated keys. The columns in the table are: Cliente, Nombre, Categoría, Usuario, and URL / IP. Each row contains a small yellow circular icon with a white letter 'L' and the text "LAB.LOCAL". The rows list the following information:

| Cliente   | Nombre                  | Categoría         | Usuario            | URL / IP  |
|-----------|-------------------------|-------------------|--------------------|---|
| LAB.LOCAL | Censys Search           | RT OSINT          | red-team@lab.local | <a href="https://search.censys.io/">https://search.censys.io/</a>     |
| LAB.LOCAL | Cobaltstrike Teamserver | RT Infrastructure |                    | 192.168.1.4:50050   |
| LAB.LOCAL | Gophish Panel           | RT Infrastructure | red-team           | <a href="https://gophish.lab.com">https://gophish.lab.com</a>         |
| LAB.LOCAL | SecurityTrails          | RT OSINT          | red-team@lab.local | <a href="https://securitytrails.com/">https://securitytrails.com/</a> |
| LAB.LOCAL | Shodan                  | RT OSINT          | red-team@lab.local | <a href="https://www.shodan.io/">https://www.shodan.io/</a>           |

At the bottom right of the interface, it says "sysPass 3.2 :: cygnux.org".

Y con esto se da por finalizada la configuración de Syspass. Obviamente, las posibilidades a realizar dentro de la aplicación son muchas y estas dependerán del tipo de infraestructura y sus componentes. Se recomienda investigar la aplicación a fondo para poder aprovechar todas sus funcionalidades.

## Servicio de Kanban - Wekan



## ¿Por qué utilizar Wekan como servicio Kanban?

### QQ Quote

Wekan es **una aplicación gratuita y de código abierto** basada en el concepto **Kanban** un término de origen japonés que literalmente significa «tarjeta» o «señalización». Este es un concepto normalmente **relacionado con el uso de tarjetas** (post-it y otros) **para indicar el progreso de los flujos de producción** en las empresas de fabricación en serie.

Wekan **está construida con el marco Meteor JavaScript** y se distribuye bajo la licencia MIT, lo que permite que cualquier persona pueda trabajar fácilmente con él y modificarlo. Wekan puede alojarse en su propio servidor con muy poco esfuerzo, garantizando que tenga todo el tiempo el control total de sus datos y pueda asegurarse de que nadie más tenga acceso a él.

Con Wekan podemos realizar una gestión de tareas pendientes en tarjetas y tableros. Las tarjetas se pueden mover entre varias Columnas. Los tableros pueden tener muchos miembros, lo que permite una fácil colaboración. [ref](#)

Poco más que añadir tras esta precisa explicación, simplemente destacar que, podemos utilizar Wekan, como sistema de gestión de tareas y progreso ante los distintos proyectos de una organización.

Por ejemplo, el equipo de Red Team, esta en pleno desarrollo de su C2 e implante, así como también tiene investigaciones abiertas en distintas materias. Para poder gestionar las tareas que surgen y las que están en curso, el Red Team Manager escoge Wekan para gestionar cada una de las tareas dentro de un Tablero asociado al departamento de Red Team. De igual forma, otros departamentos consiguen la misma distribución de tareas creando sus propios tableros y tarjetas.

En los siguientes apartados se verá como se despliega y configura la aplicación web de Wekan, así como se expondrán ciertos ejemplos prácticos de uso.

## Despliegue de contenedor de Wekan

Comenzamos el despliegue del contenedor de Wekan con el siguiente archivo de configuración docker-compose:

```
version: '2'
services:
  wekandb:
    image: mongo:5
    container_name: wekan-db
    restart: always
    command: mongod --logpath /dev/null --oplogSize 128 --quiet
    networks:
      - wekan-tier
    expose:
      - 27017
  volumes:
    - /etc/localtime:/etc/localtime:ro
    - /etc/timezone:/etc/timezone:ro
    - /srv/data/wekan/wekan-db:/data/db
    - /srv/data/wekan/wekan-db-dump:/dump
  wekan:
    image: quay.io/wekan/wekan
    container_name: wekan-app
    restart: always
    networks:
      - wekan-tier
    ports:
      - "127.0.0.1:8003:8080"
    environment:
      - WRITABLE_PATH=/data
      - MONGO_URL=mongodb://wekandb:27017/wekan
      - ROOT_URL=https://wekan.lab.local
      - WITH_API=true
      - RICHER_CARD_COMMENT_EDITOR=false
      - CARD_OPENED_WEBHOOK_ENABLED=false
      - BIGEVENTS_PATTERN=NONE
      - BROWSER_POLICY_ENABLED=true
    depends_on:
      - wekandb
  volumes:
    - /etc/localtime:/etc/localtime:ro
    - /srv/data/wekan/wekan-files:/data:rw
volumes:
  wekan-files:
    driver: local
  wekan-db:
    driver: local
  wekan-db-dump:
    driver: local
```

```
networks:
  wekan-tier:
    driver: bridge
    external: false
    driver_opts:
      com.docker.network.bridge.name: br_wekan
```

### Note

Es necesario definir algunos campos en el archivo de configuración, por ejemplo, el campo ROOT\_URL indicando la URL de acceso al wekan dentro de la organización.

Este archivo YAML, lo vamos a situar en la ruta: `/srv/containers/wekan/docker-compose.yml`. Antes de desplegar el contendor, creamos la ruta del volumen en el directorio de datos:

```
mkdir -p /srv/data/wekan/
```

Pero, antes de desplegar el contenedor, tenemos que dar permisos al directorio donde wekan guardará los datos. De otra forma, ocurrirá un error de permission denied:

```
mkdir -p /srv/data/wekan/wekan-files
chmod 777 /srv/data/wekan/wekan-files
```

Para proceder a su despliegue simplemente ejecutamos el siguiente comando:

```
docker compose up -d
```

Tras el despliegue podemos ver como el contenedor esta corriendo con:

```
docker compose ps
```

```
root@vps:/srv/containers/wekan# docker compose ps
NAME           COMMAND            SERVICE          STATUS        PORTS
wekan-app      "bash -c 'ulimit -s ..."   wekan        running
wekan-db       "docker-entrypoint.s..."   wekandb     running
root@vps:/srv/containers/wekan#
```

Se añade el subdominio al `/etc/hosts` para gestionar el servicio (como se ha hecho anteriormente) quedando el archivo así:

```

127.0.0.1 localhost
127.0.1.1 vps

# The following lines are desirable for IPv6 capable hosts
::1      ip6-localhost ip6-loopback
fe00::0  ip6-localnet
ff00::0  ip6-mcastprefix
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters

# LAB.LOCAL DNS records
192.168.10.1 vps.lab.local
192.168.10.2 laptop.lab.local

192.168.10.1 git.lab.local
192.168.10.1 md.lab.local
192.168.10.1 chat.lab.local
192.168.10.1 pass.lab.local
192.168.10.1 wekan.lab.local

```

También se crea el archivo de configuración de Nginx para gestionar el virtual host. Este archivo se sitúa en `/etc/nginx/sites-available/wekan.lab.local` y tiene el siguiente contenido:

```

server {
    listen 192.168.10.1:80;
    server_name wekan.lab.local;
    # Redirect all non-https requests
    rewrite ^ https://$host$request_uri? permanent;
}

server {
    listen 192.168.10.1:443 ssl http2;

    include snippets/self-signed.conf;
    include snippets/ssl-params.conf;

    server_name wekan.lab.local;

    client_max_body_size 0;
    proxy_http_version 1.1;
    proxy_request_buffering off;

    location / {

```

```
proxy_pass http://127.0.0.1:8003;
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
}

}
```

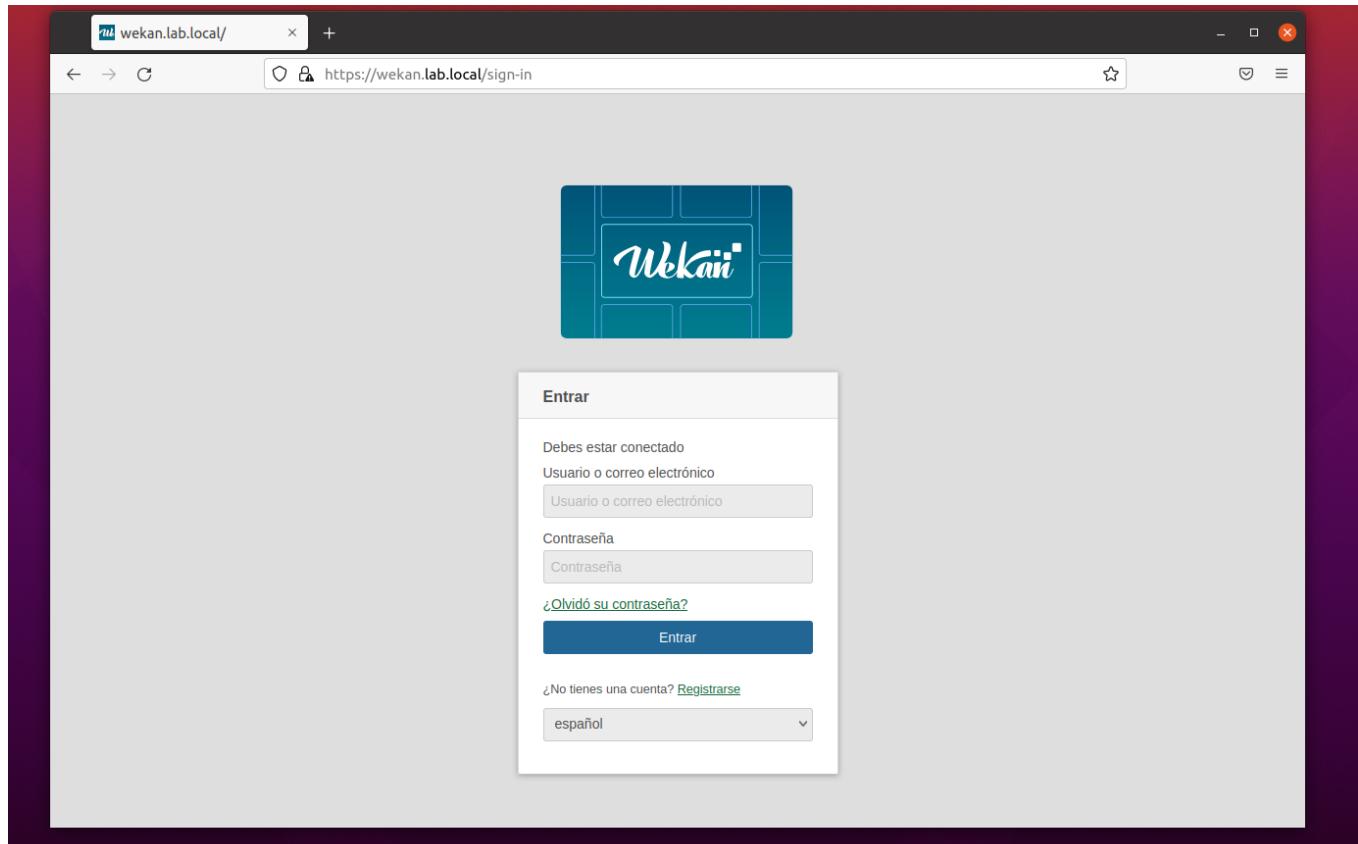
Tras esto, se crea el enlace simbólico para activarlo:

```
ln -s /etc/nginx/sites-available/wekan.lab.local /etc/nginx/sites-enabled/wekan.lab.local
```

Se comprueba que todo esta correcto y se reinicia el servicio de Nginx y Dnsmasq para aplicar los cambios:

```
nginx -t
systemctl restart nginx
systemctl restart dnsmasq
```

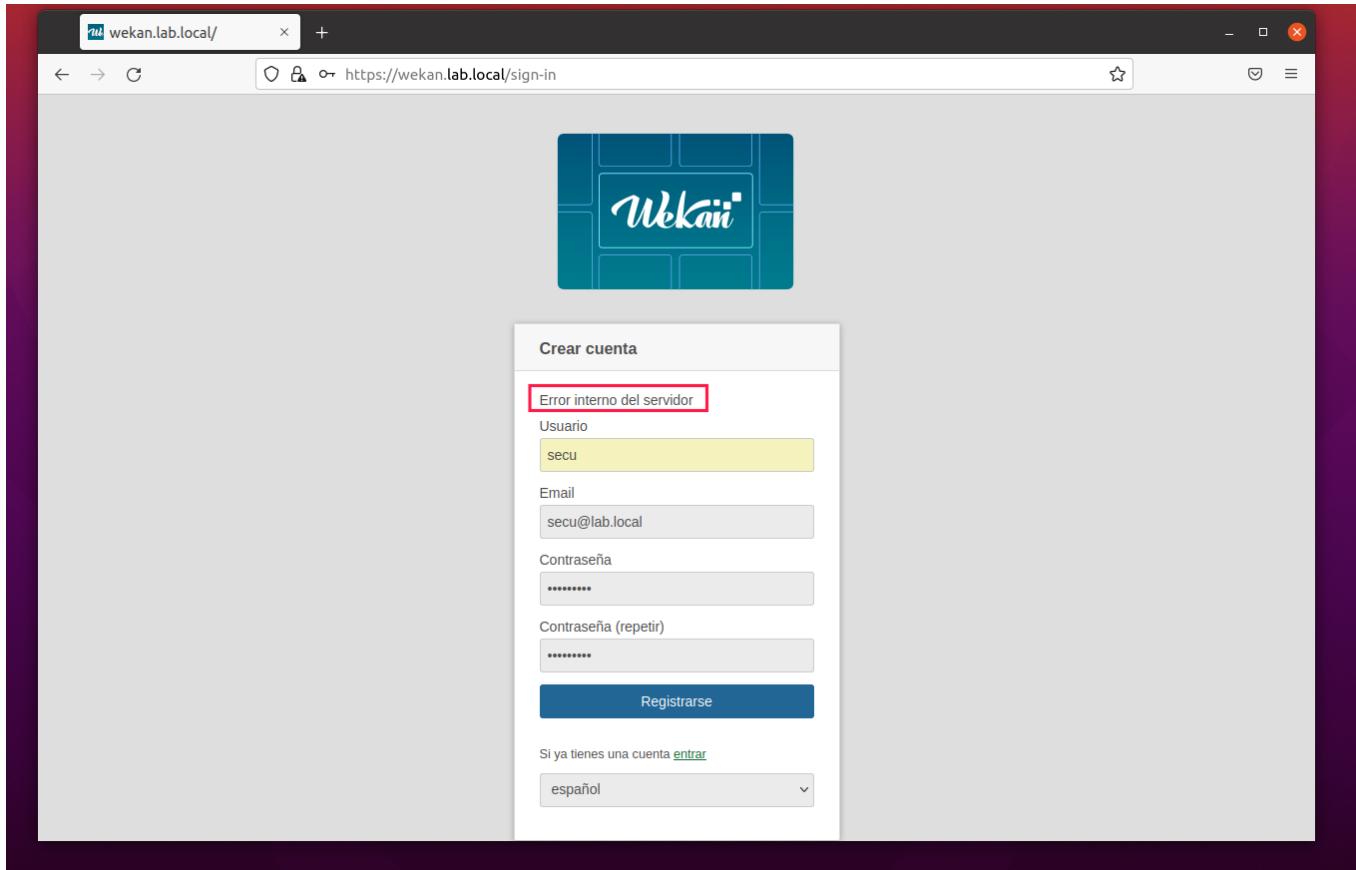
Y, desde la máquina **LAPTOP**, visitamos la dirección <https://wekan.lab.local> (tras excepcionar el certificado SSL) encontramos:



# Configuración e instalación de Wekan

En este momento podemos proceder a registrarnos haciendo click en el link de "Registrarse".

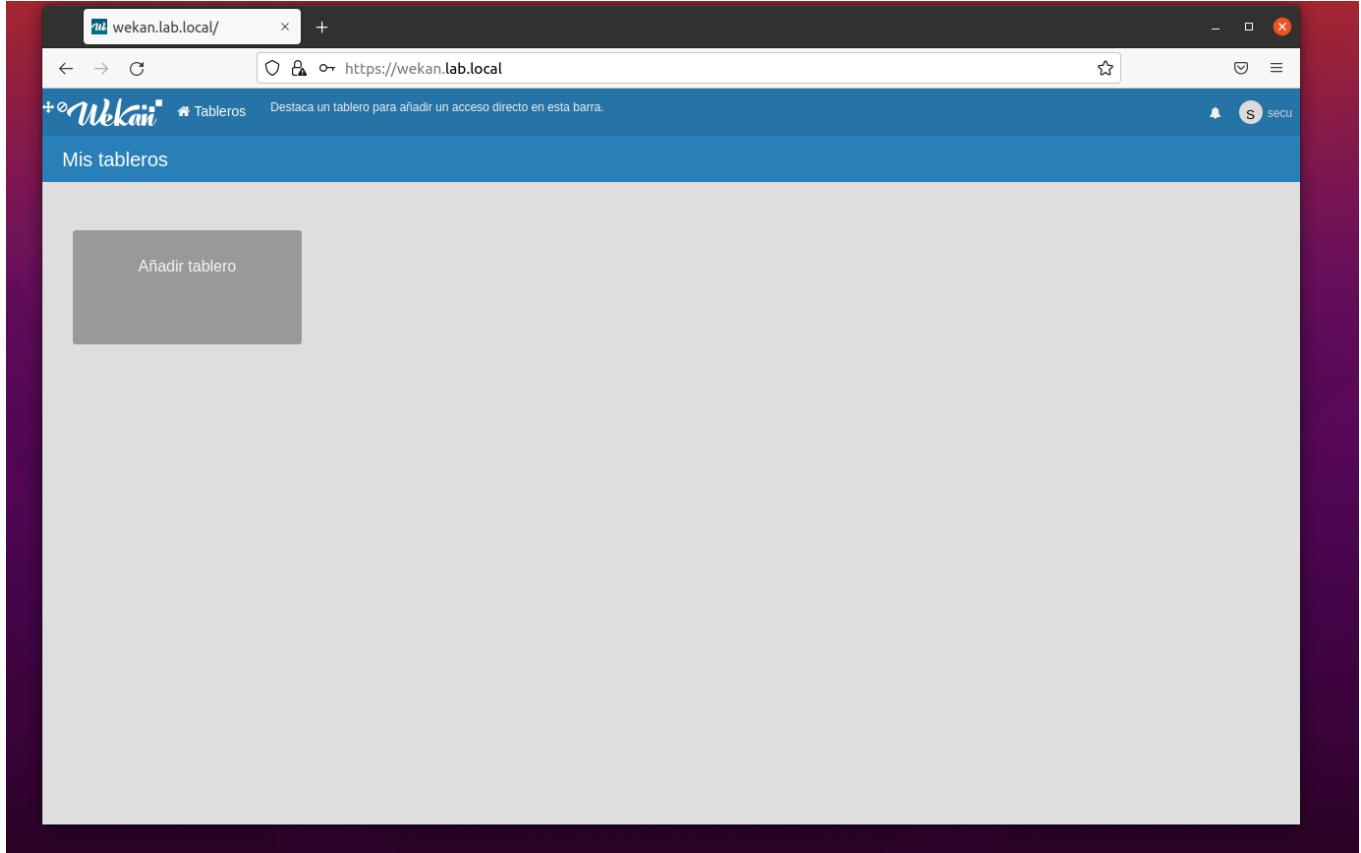
Al hacerlo, cuando completamos el registro, aparece **un error interno del servidor**.



## Note

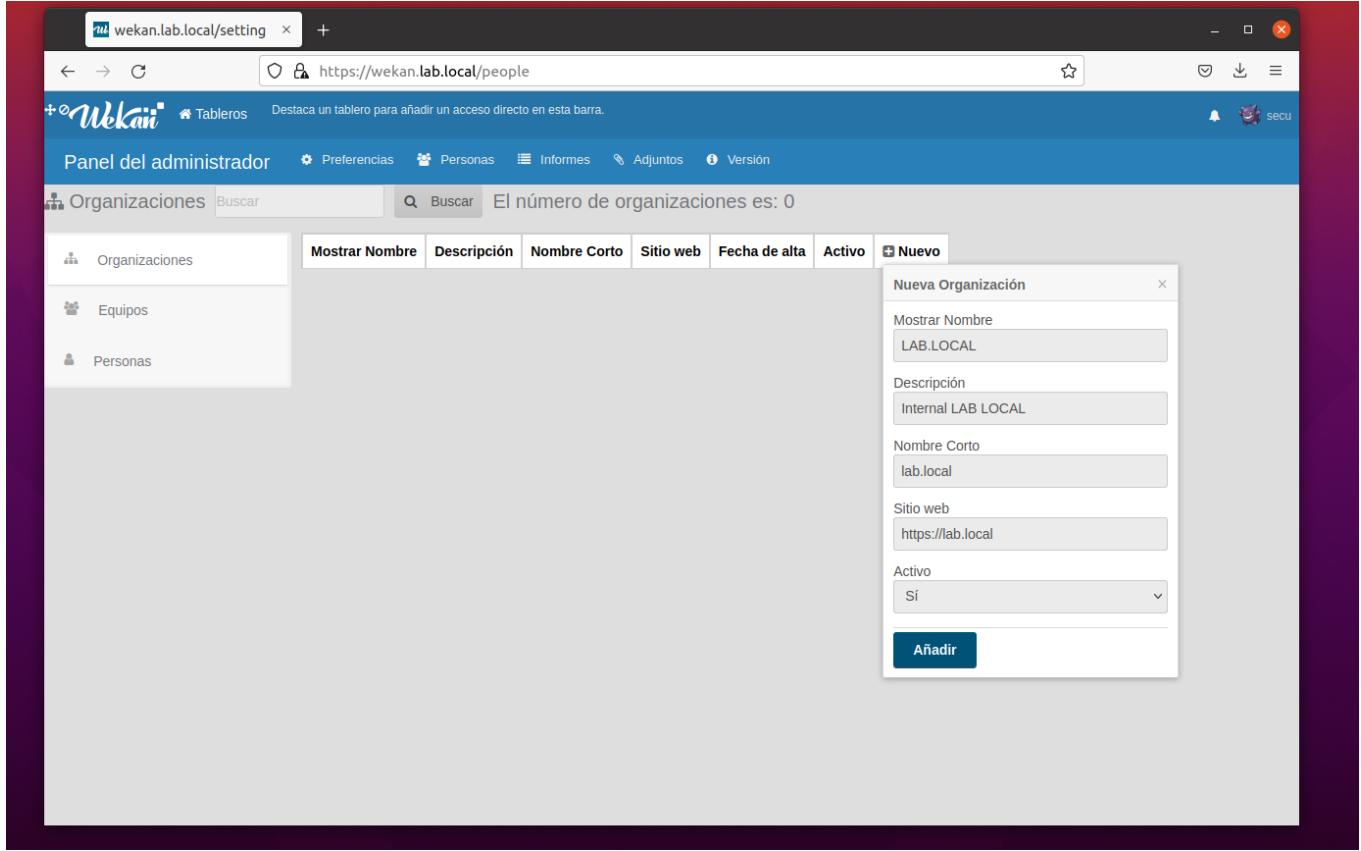
Aunque recibamos este error en el registro, el usuario se ha creado con éxito. Este error se debe a que no hemos especificado una configuración del servidor SMTP a utilizar en Wekan. En nuestro caso, como no vamos a utilizar un servidor SMTP ignoraremos este error.

Para acceder simplemente hacemos click en el link de "Entrar" y procedemos a iniciar sesión (ya que el usuario ha sido creado exitosamente):



Al crear el primer usuario, se nos otorga el rol de **Administrador**, lo que significa que procederemos a realizar una serie de configuraciones y creación de los elementos necesarios para comenzar a utilizar Wekan.

Primero, procederemos a crear una nueva **Organización** como se hizo en anteriores servicios.



Una vez definida la organización (en este caso simplemente se creará la organización interna que vamos a gestionar). El siguiente paso es crear los equipos que van a trabajar dentro de la organización.

#### Note

Estos pasos son similares a los que realizamos en la configuración del servicio Syspass, ya que en Wekan realizaremos una gestión y organización de tareas (en lugar de credenciales), pero mucho más básica.

The screenshot shows the Wekan administrator interface at <https://wekan.lab.local/people>. The top navigation bar includes links for Tableros, Preferencias, Personas, Informes, Adjuntos, and Versión. A sidebar on the left lists Organizaciones, Equipos, and Personas. The main content area displays a table header with columns: Mostrar Nombre, Descripción, Nombre Corto, Sitio web, Fecha de alta, Activo, and Nuevo. A modal window titled "Nuevo Equipo" is open, prompting for "Mostrar Nombre" (RED TEAM), "Descripción" (Red Team Operators), "Nombre Corto" (redteam), "Sitio web" (empty), and "Activo" (Sí). A blue "Añadir" button is at the bottom of the modal.

Un ejemplo de los diferentes equipos creados podría ser el siguiente:

The screenshot shows the Wekan administrator interface displaying a list of three teams. The table has columns: Mostrar Nombre, Descripción, Nombre Corto, Sitio web, Fecha de alta, Activo, and actions (Editar, ...). The teams listed are:

| Mostrar Nombre  | Descripción               | Nombre Corto | Sitio web   | Fecha de alta               | Activo | Acción                              |
|-----------------|---------------------------|--------------|---|-----------------------------|--------|-------------------------------------|
| BLUE TEAM       | Blue Team Analysts        | blueteam     | <a href="https://blue-team.lab.local">https://blue-team.lab.local</a> | 29 de octubre de 2022 18:45 | Sí     | <input type="checkbox"/> Editar ... |
| Infraestructure | Sysadmins Infraestructure | infra        | <a href="https://internal.lab.local">https://internal.lab.local</a>   | 29 de octubre de 2022 18:45 | Sí     | <input type="checkbox"/> Editar ... |
| RED TEAM        | Red Team Operators        | redteam      | <a href="https://red-team.lab.local">https://red-team.lab.local</a>   | 29 de octubre de 2022 18:44 | Sí     | <input type="checkbox"/> Editar ... |

También procedemos a crear los primeros usuarios:

The screenshot shows the Wekan web application interface. On the left, there's a sidebar with 'Equipos' and 'Personas' sections. The main area displays a table with columns: 'Nombre', 'Apellido', 'Nombre de usuario', 'Activo', 'Fecha de creación', 'Administrador', and 'Contraseña'. One row is selected, showing 'secu' in the 'Nombre' column. A modal window titled 'Nuevo usuario' is open on the right, containing fields for 'Nombre completo' (set to 'Vici'), 'Nombre de usuario' (set to 'vici'), 'Iniciales' (set to 'vici'), 'Correo electrónico' (set to 'vici@lab.local'), 'Importar Usuarios' (set to 'secu'), 'Administrador' (set to 'No'), 'Activo' (set to 'Sí'), 'Tipo de autenticación' (set to 'Contraseña'), 'Organizaciones' (set to 'LAB.LOCAL'), 'Equipos' (set to 'RED TEAM'), and 'Contraseña' (represented by a series of asterisks). A blue 'Añadir' button is at the bottom of the modal.

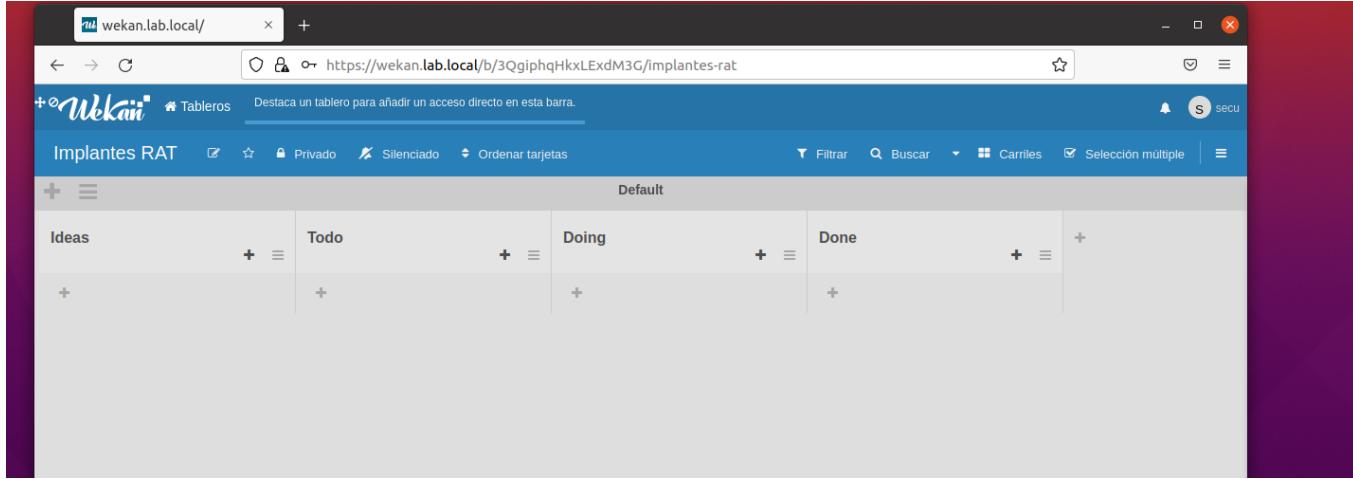
Llegados a este punto, podemos comenzar con la creación de tableros y empezar a organizar las tareas de la infraestructura.

#### Note

Los **tableros** nos permiten gestionar las tareas a realizar dentro de una organización. Cada tablero podría abordar todas las tareas relacionadas con una actividad o equipo.

Por ejemplo, el equipo de Red Team de una empresa, podría tener un tablero para agrupar las tareas relacionadas con el diseño y desarrollo de su Command and Control e Implantes.

Para ejemplificar esta situación, procedemos a crear un tablero llamado "Implantes RAT". Dentro de este tablero comenzamos a definir las columnas en las que repartiremos las tareas. Puede quedar algo como en la siguiente imagen:

A screenshot of a web-based Kanban board titled "Implantes RAT". The board has four columns: "Ideas", "Todo", "Doing", and "Done". Each column contains a single card with a plus sign to add more. The top navigation bar includes links for "Tableros", "Privado", "Silenciado", "Ordenar tarjetas", "Filtrar", "Buscar", "Carillas", "Selección múltiple", and a user icon.

### Note

Se recomienda estructurar el tablero con las estas columnas o similar debido a que permiten tener una pseudo-organización basada en el tiempo y en el estado de las tareas.

En el tablero se pueden ir creando **Tarjetas**, que serían algo así como las "tareas". Algunas de las acciones que se puede realizar sobre estas tarjetas son:

- Asignar un título y descripción
- Cambiarle el color
- Añadir tags, fechas de vencimiento o de entrada
- Asignarlas a usuarios existentes
- Poner comentarios dentro de la tarjeta
- Crear listas de verificaciones (sería como subtareas que, cuando completas, van marcándose los checks)
- Y muchas otras funcionalidades

Un ejemplo de como un equipo podría ir organizando las tareas a realizar, es el siguiente:

Con esto finalizamos la configuración y uso básico de Wekan. Aunque es posible realizar muchas más acciones y sacarle mucho más partido a la plataforma, pero en esta guía no profundizaremos más.

## Servicio de Bitácoras - WikiJS



### ¿Por qué utilizar WikiJS?

WikiJS es una aplicación de bitácoras de código abierto. Esta desarrollada en Nodejs y combina el uso de Git y Markdown. Es bastante ligero (en comparación con otros servicios de Wiki) y tiene una serie de características que lo hacen preferible a otras alternativas:

- WikiJS incorpora un editor visual muy cómodo, estilo Hackmd, que permite escribir el contenido y verlo renderizado para hacerse una idea del resultado final.
- Todo el contenido se guarda en archivos Markdown, y es posible, sincronizar este contenido con nuestro propio repositorio de control de versiones.
- WikiJS se integra con multitud de servicios, en la [sección de módulos de su web oficial](#), se pueden ver todas las integraciones son servicios de distintas categorías y propósitos. Algunos de interés son: Google, Azure, Github o LDAP (como proveedores de autenticación alternativos), también AWS, Dropbox o Azure como servicios de almacenamiento, entre muchos otros.
- La instalación es sencilla, la [documentación de su página web oficial](#) es muy precisa y completa. Lo que facilita cualquier problema o duda que pueda surgir durante el despliegue o uso.
- La vista de la documentación creada es limpia y sigue un patrón de diseño que facilita la búsqueda y estructuración de la información contenida.

Realmente, es un servicio muy maduro y cuando se indaga en la documentación o en el propio despliegue uno aprecia el trabajo y esfuerzo que han hecho posible desarrollar esta herramienta.

Y, aunque en esta guía, no vamos a cubrir más que el contenido más superficial de todo lo que se puede extraer de este software, considero que es el servicio que toda empresa debería de tener como bitácora o wiki interna.

## Despliegue de contenedor de WikiJS

Comenzamos el despliegue del contenedor de WikiJS con el siguiente archivo de configuración docker-compose:

```
version: "3"
services:
  db:
    image: postgres:11-alpine
    environment:
      POSTGRES_DB: wiki
      POSTGRES_PASSWORD: PgP4ssw0rd
      POSTGRES_USER: wikijs
    logging:
      driver: "none"
    restart: unless-stopped
    volumes:
      - /srv/data/wikijs/db-data:/var/lib/postgresql/data
  wiki:
    image: ghcr.io/requarks/wiki:2
```

```
depends_on:
  - db
environment:
  DB_TYPE: postgres
  DB_HOST: db
  DB_PORT: 5432
  DB_USER: wikijs
  DB_PASS: PgP4ssw0rd
  DB_NAME: wiki
restart: unless-stopped
ports:
  - "127.0.0.1:8004:3000"
```

### Note

Es necesario cambiar la contraseña del usuario de la base de datos que se ha definido en este YAML. También es posible cambiar el nombre del usuario y base de datos si se requiere.

Este archivo YAML, lo vamos a situar en la ruta: `/srv/containers/wikijs/docker-compose.yml`. Antes de desplegar el contenedor, creamos la ruta del volumen en el directorio de datos:

```
mkdir -p /srv/data/wikijs/
```

Para proceder a su despliegue simplemente ejecutamos el siguiente comando:

```
docker compose up -d
```

Tras el despliegue podemos ver como el contenedor esta corriendo con:

```
docker compose ps
```

```
root@vps:/srv/containers/wikijs# docker compose ps
NAME           COMMAND            SERVICE      STATUS        PORTS
wikijs-db-1    "docker-entrypoint.s..."  db          running      5432/tcp
wikijs-wiki-1   "docker-entrypoint.s..."  wiki        running      3443/tcp, 127.0.0.1:8004->3000/tcp
root@vps:/srv/containers/wikijs#
```

Se añade el subdominio al `/etc/hosts` para gestionar el servicio (como se ha hecho anteriormente) quedando el archivo así:

```
127.0.0.1 localhost
127.0.1.1 vps
```

```

# The following lines are desirable for IPv6 capable hosts
::1      ip6-localhost ip6-loopback
fe00::0  ip6-localnet
ff00::0  ip6-mcastprefix
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters

# LAB.LOCAL DNS records
192.168.10.1 vps.lab.local
192.168.10.2 laptop.lab.local

192.168.10.1 git.lab.local
192.168.10.1 md.lab.local
192.168.10.1 chat.lab.local
192.168.10.1 pass.lab.local
192.168.10.1 wekan.lab.local
192.168.10.1 wiki.lab.local

```

También se crea el archivo de configuración de Nginx para gestionar el virtual host. Este archivo se sitúa en `/etc/nginx/sites-available/wiki.lab.local` y tiene el siguiente contenido:

```

server {
    listen 192.168.10.1:80;
    server_name wiki.lab.local;
    # Redirect all non-https requests
    rewrite ^ https://$host$request_uri? permanent;
}

server {
    listen 192.168.10.1:443 ssl http2;

    include snippets/self-signed.conf;
    include snippets/ssl-params.conf;

    server_name wiki.lab.local;

    client_max_body_size 0;
    proxy_http_version 1.1;
    proxy_request_buffering off;

    location / {
        proxy_pass http://127.0.0.1:8004;
        proxy_set_header Host $host;
    }
}

```

```
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

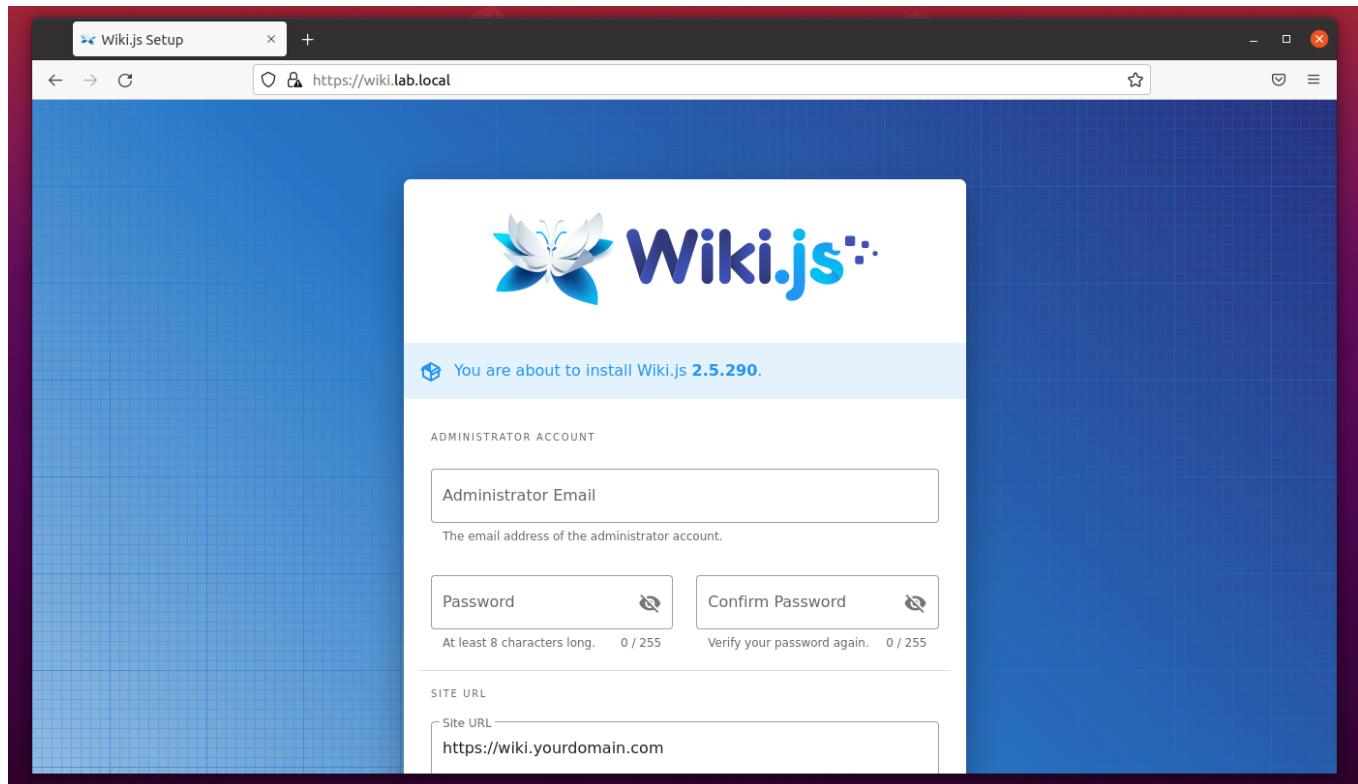
Tras esto, se crea el enlace simbólico para activarlo:

```
ln -s /etc/nginx/sites-available/wiki.lab.local /etc/nginx/sites-
enabled/wiki.lab.local
```

Se comprueba que todo esta correcto y se reinicia el servicio de Nginx y Dnsmasq para aplicar los cambios:

```
nginx -t
systemctl restart nginx
systemctl restart dnsmasq
```

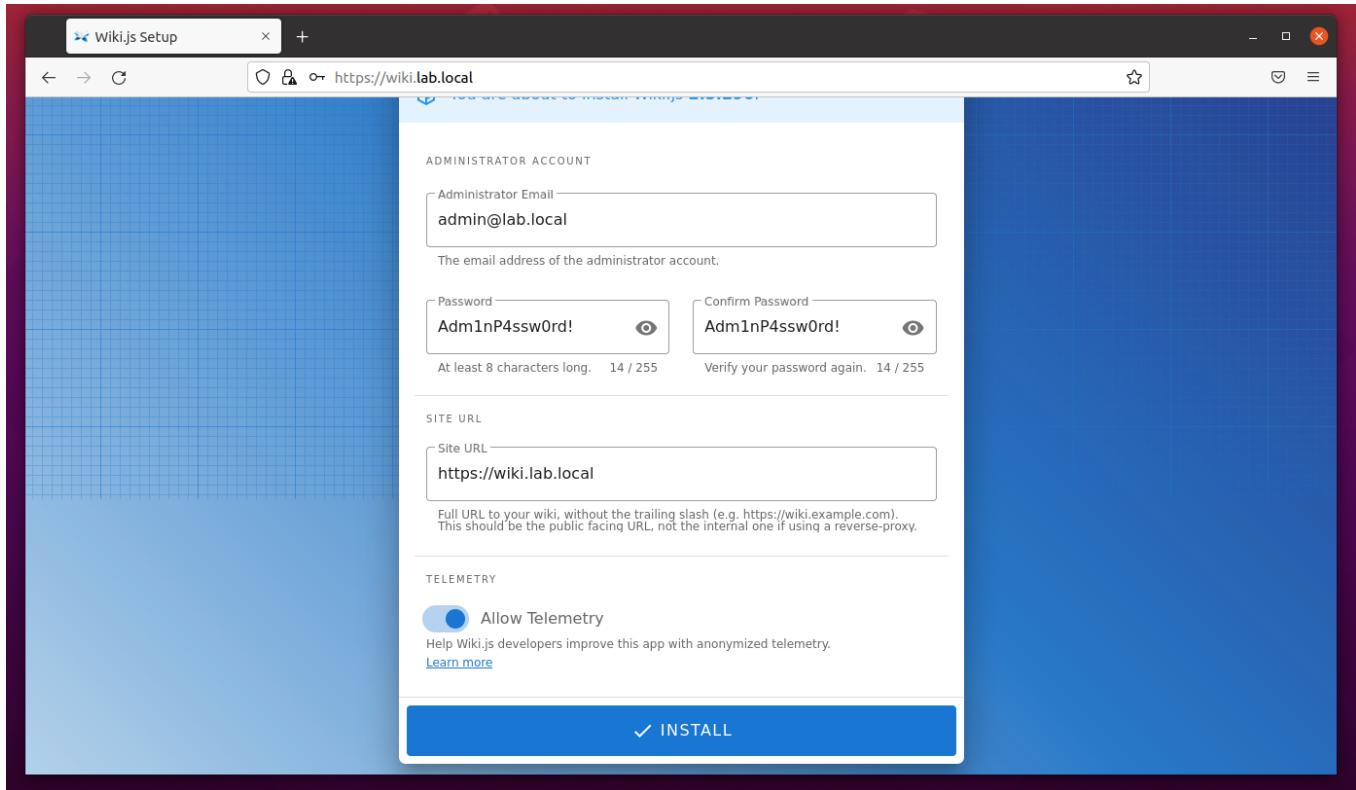
Y, desde la máquina **LAPTOP**, visitamos la dirección <https://wiki.lab.local> (tras excepcionar el certificado SSL) encontramos:



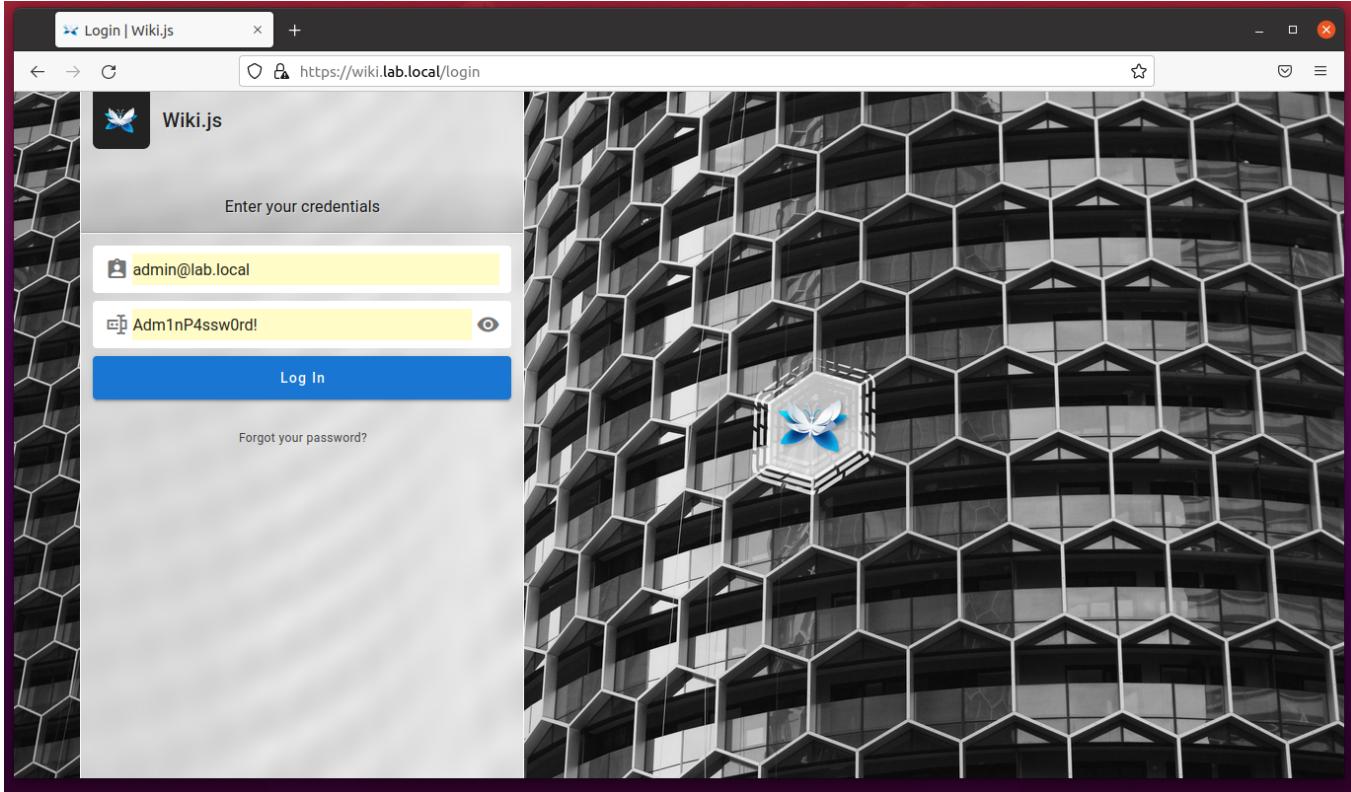
# Configuración de instalación de WikiJS

Al acceder por primera vez, WikiJS solicita realizar la instalación inicial del sitio. Para ello, simplemente introducimos las credenciales del que será el administrador de la Wiki, así como el subdominio utilizado para acceder a la misma:

- Email Administrador: `admin@lab.local`
- Contraseña: `Adm1nP4ssword!`
- URL: `https://wiki.lab.local`



Tras esto, se completa el registro, y en la siguiente página, procedemos a hacer login utilizando las credenciales administrativas:



WikiJS permite realizar gran cantidad de configuraciones en el sitio, como definir diferentes métodos de autenticación, integración con servicios en nube, extensiones, utilidades a incorporar, etc.

En esta guía, veremos una parte muy reducida de todas las funciones que se pueden utilizar en WikiJS, ya que dependiendo del objetivo con el que se despliegue la infraestructura, se elegirá añadir una serie de configuraciones adicionales o no.

En primer lugar comenzamos con la creación de grupos dentro de la plataforma. Estos grupos, al igual que en las configuraciones vistas hasta ahora, se destinarán a plasmar los diferentes departamentos o grupos, dentro de la organización. En nuestro caso, procederemos a crear tres: **Red Team**, **Blue Team** e **Infraestructura**.

En la sección administrativa, simplemente procedemos a crear los grupos quedando de la siguiente forma:

The screenshot shows the Wiki.js Administration Area with the URL <https://wiki.lab.local/a/groups>. The left sidebar has a 'Navigation' section with links like Pages, Tags, Theme, Users, Groups, and more. The main content area is titled 'Groups' with the sub-instruction 'Manage groups and their permissions'. It lists five groups with the following details:

| ID | Name           | Users | Created          | Last Updated     |
|----|----------------|-------|------------------|------------------|
| 1  | Administrators | 1     | Today at 8:15 AM | Today at 8:15 AM |
| 2  | Guests         | 1     | Today at 8:15 AM | Today at 8:15 AM |
| 3  | Red Team       | 0     | Today at 8:29 AM | Today at 8:29 AM |
| 4  | Blue Team      | 0     | Today at 8:31 AM | Today at 8:31 AM |
| 5  | Infrastructure | 0     | Today at 8:31 AM | Today at 8:31 AM |

At the bottom right of the main content area is a blue button labeled '+ NEW GROUP'.

A continuación, creamos un par de usuarios y los vinculamos a cada grupo:

The screenshot shows the Wiki.js Administration Area with the URL <https://wiki.lab.local/a/users>. The left sidebar has a 'Navigation' section with links like Pages, Tags, Theme, Users, Groups, and more. A modal dialog is open titled '+ New User' with the following fields filled in:

- Provider: Local
- Email Address: secu@lab.local
- Temporary Password: P4ssw0rd!
- Name: Security
- Assign to Group(s): Red Team, Infrastructure

Below the form, there are two checkboxes:

- Require password change on first login
- Send a welcome email

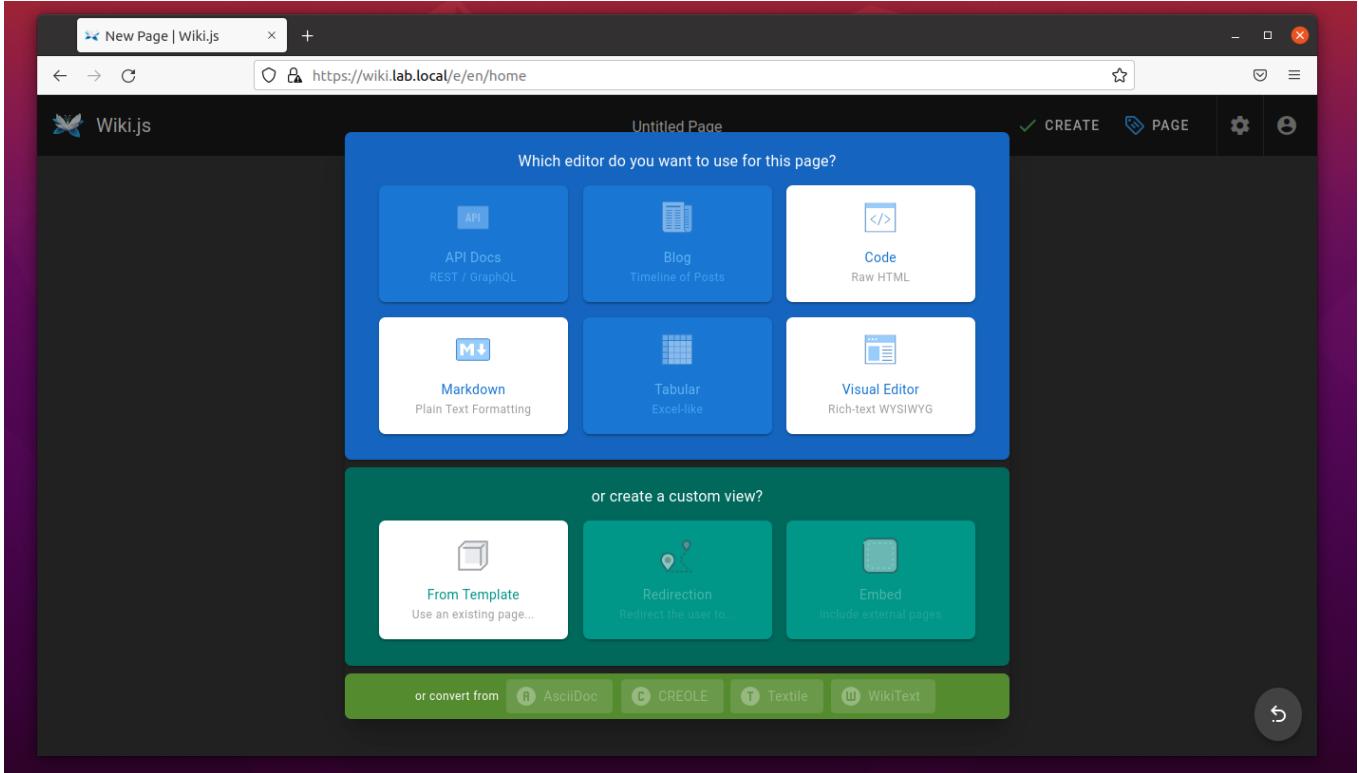
The screenshot shows the 'Users' page in the Wiki.js Administration Area. The left sidebar has a 'Navigation' section with 'Pages' (0), 'Tags' (0), 'Theme', 'Users', 'Groups' (2), and 'Users' (2) selected. Other sections include 'Modules', 'Analytics', 'Authentication', 'Comments', 'Editor', 'Extensions', and 'Logging'. The main content area is titled 'Users' and 'Manage users'. It features a search bar with 'Search Users...' and a dropdown for 'All Providers'. A table lists six users: Guest (guest@example.com, Local, 44 minutes ago, Never), Administrator (admin@lab.local, Local, 44 minutes ago, 8 minutes ago), Secury (secu@lab.local, Local, 2 minutes ago, Never), Vici (vici@lab.local, Local, a minute ago, Never), Jack (jack@lab.local, Local, a few seconds ago, Never), and Gygas (gygas@lab.local, Local, a few seconds ago, Never). Each user row has a lock icon and a green circular status indicator.

| ID | Name          | Email             | Provider | Created           | Last Login    |
|----|---------------|-------------------|----------|-------------------|---------------|
| 2  | Guest         | guest@example.com | Local    | 44 minutes ago    | Never         |
| 1  | Administrator | admin@lab.local   | Local    | 44 minutes ago    | 8 minutes ago |
| 3  | Secury        | secu@lab.local    | Local    | 2 minutes ago     | Never         |
| 4  | Vici          | vici@lab.local    | Local    | a minute ago      | Never         |
| 5  | Jack          | jack@lab.local    | Local    | a few seconds ago | Never         |
| 6  | Gygas         | gygas@lab.local   | Local    | a few seconds ago | Never         |

La estructura de páginas que vamos a seguir es la siguiente:

- OSINT
- WEB
- PERSISTENCE
  - Windows Persistence
  - Linux Persistence
- CREDENTIAL ACCESS
  - Windows Credential Access
  - Linux Credential Access

Primero comenzamos creando una nueva página. Elegimos como editor: **Markdown**



Se procede a crear la página con el editor de Markdown, dónde se puede ver el resultado renderizado en tiempo real:

A screenshot of the Wiki.js editor interface. The title bar says 'Edit OSINT | Wiki.js'. The address bar shows 'https://wiki.lab.local/e/en/home'. The main area has a toolbar with various editing icons. On the left is the Markdown code editor containing the following text:

```
1 # OSINT
2
3 OSINT Guide and Resources
4
5 > An info text box
6 {.is-info}
7
8 <kbd>test</kbd>
9
10 ``python
11 print("abc")
12 ...
```

On the right is the rendered preview pane, which displays:

## OSINT

OSINT Guide and Resources

An info text box

test

```
1 | print("abc")
```

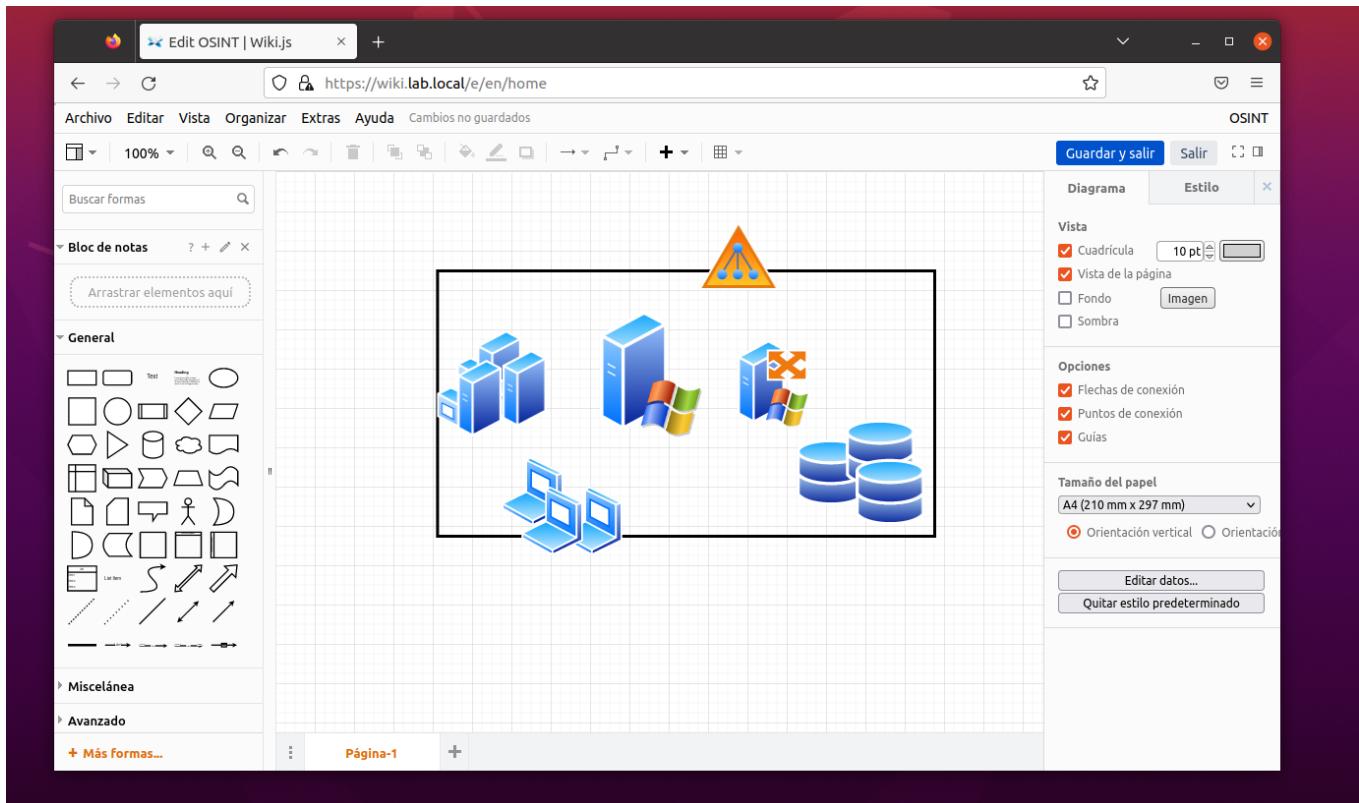
The status bar at the bottom shows 'EN /home Markdown Ln 12, Col 4'.

Una vez se finaliza la página, esta aparece en la sección principal de Wikijs. Finalmente tras crear las diferentes páginas citadas anteriormente, se obtiene un resultado como el siguiente:

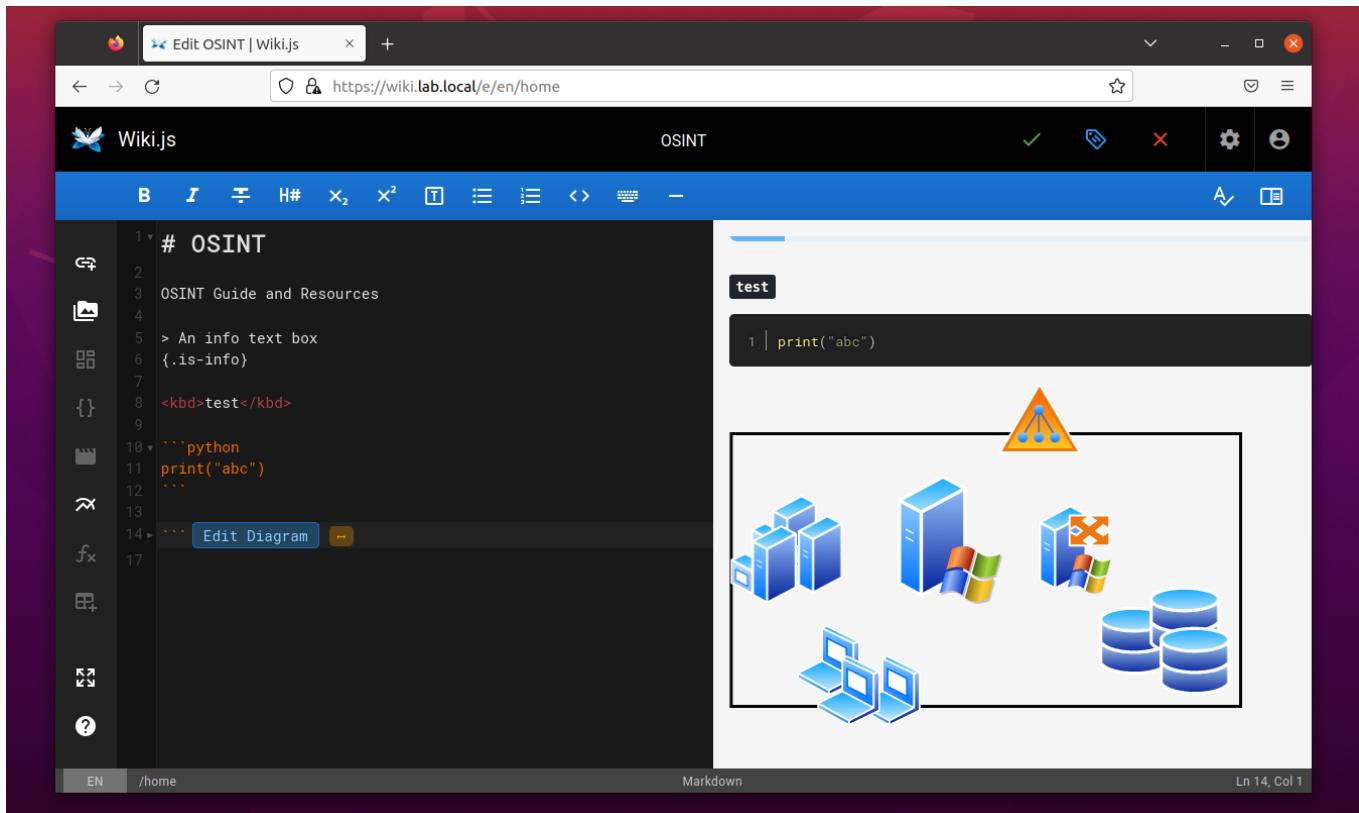
The screenshot shows a browser window with the URL <https://wiki.lab.local>. The title bar says "OSINT | Wiki.js". The main content area displays a page titled "OSINT" under the heading "OSINT Guide and Resources". On the left, there is a sidebar with a blue header containing links: "Credential Access", "Persistence", "OSINT" (which is currently selected), and "Web Exploitation". The main content area includes a search bar at the top right, a "Comments" section with a text input field and a blue "Write comment" button, and a code editor section with the text "1 | print('abc')". There is also a blue info box with the text "An info text box". The overall interface is clean and modern.

The screenshot shows a browser window with the URL <https://wiki.lab.local/a/pages/visualize>. The title bar says "Admin | Wiki.js". The main content area is titled "Visualize Pages" and describes it as a "Dendrogram representation of your pages". It features a hierarchical tree diagram where the root node "en" branches into "Persistence", "Web Exploitation", and "Credential Access", which further branches into "OSINT" and "Linux Persistence", "Windows Persistence", "Linux Credential Access", and "Windows Credential Access". On the left, there is a sidebar titled "ADMINISTRATION AREA" with sections for "Dashboard", "Site", "General", "Locale", "Navigation", "Pages" (with 8 items), "Tags" (with 1 item), "Theme", "Users", "Groups" (with 5 items), "Users" (with 6 items), "Modules", and "Analytics". The "Pages" section is currently active. The bottom right corner of the main content area says "Powered by Wiki.js".

Para finalizar, otra opción de interés es la posibilidad de crear un diagrama desde la vista de creación/edición de páginas. Si hacemos click sobre el botón que pone "Insert Diagram", situado en la sección izquierda. Se nos abre una aplicación embebida que no es otra que Drawio:



Cuando acabamos de diseñar el diagrama, podemos importarlo directamente a la página haciendo click en el botón de "Guardar y salir":



### Note

Debido a que WikiJS incorpora el propio Drawio, nos ahorraremos realizar el despliegue en esta guía. No obstante, los archivos necesarios para realizar un despliegue por separado se han adjuntado a la carpeta de archivos.

Y, tras finalizar la creación de páginas, el resto se deja para investigación propia de la aplicación. Existen muchas posibilidades de organización, customización y personalización tanto de las páginas como de la visualización de las mismas. Te animo a profundizar en ello!

En la siguiente sección, se realizará la configuración e instalación, del servicio de videoconferencias gratuito: Jitsi Meet.

## Servicio de Videoconferencias - Jitsi Meet



# Jitsi Meet

Jitsi Meet es una solución de videoconferencias y llamadas gratuita, de código abierto y que se puede montar on-premises.

- Es posible compartir pantalla y presentaciones (entre otras cosas).
- Se puede invitar a usuarios externos a una videollamada a través de una url personalizada (si, jitsi permite la identificación de salas en base a urls personalizadas).
- Dispone de un chat integrado en el servicio que permite escribir durante las videollamadas.

- También permite la posibilidad de editar documentos de forma conjunta (pero esta funcionalidad ya la hemos cubierto con Hackmd).
- Dispone de aplicación móvil (Android y iOS) y esta soportado en todos los navegadores web.
- Finalmente, también existe una solución desplegada en la nube que se puede utilizar si, por el contrario, no se quiere alojar on-premises o no interesa desplegarlo en su infraestructura. Si se desea probar se puede encontrar visitando la URL:  
<https://meet.jit.si/>

El despliegue de esta aplicación web no es sencillo, y algunas características no me han funcionado en las ultimas versiones. No obstante, se ha decidido utilizar este servicio, porque es de los pocos que hay que permiten alojar en tu propia infraestructura (y mantener el control) de un software de videoconferencias gratuito.

Y es por ello que, a continuación, se explicarán los pasos a seguir para poder desplegar Jitsi Meet y utilizarlo dentro de la Infraestructura.

## Despliegue de contenedor Jitsi Meet

Para el despliegue utilizaremos la versión estable más reciente de las que se pueden encontrar en su [repositorio oficial](#). En este momento utilizaré la release: **stable-7882**.



Utilizamos una release estable, porque con otras releases (las que vienen en la documentación oficial), dan problemas de conexión y no nos ha funcionado bien el despliegue. En cambio con las estables, si.

Primero instalaremos el paquete unzip (necesario para descomprimir la release):

```
apt install -y unzip
```

Después, nos descargaremos la release elegida de Github sobre la carpeta de contenedores:

```
cd /srv/containers/
wget https://github.com/jitsi/docker-jitsi-meet/archive/refs/tags/stable-
7882.zip
```

Tras esto, renombramos el directorio y eliminamos el zip de la release:

```
mv docker-jitsi-meet-stable-7882/ jitsi
rm stable-7882.zip
cd jitsi
```

Una vez estamos dentro del directorio de jitsi (`/srv/containers/jitsi`), procedemos a copiar el archivo `env.example` a `.env`, este archivo es el que contendrá toda la información relativa a nuestra instalación de Jitsi.

```
cp env.example .env
```

Antes de editar el archivo `env`, procedemos a reemplazar el contenido del archivo `docker-compose.yml` que trae la release por el siguiente contenido:

```
version: '3.5'

services:
  # Frontend
  web:
    image: jitsi/web:${JITSI_IMAGE_VERSION:-stable-7882}
    restart: ${RESTART_POLICY:-unless-stopped}
    ports:
      - '127.0.0.1:${HTTPS_PORT}:443'
    volumes:
      - ${CONFIG}/web:/config:Z
      - ${CONFIG}/web/crontabs:/var/spool/cron/crontabs:Z
      - ${CONFIG}/transcripts:/usr/share/jitsi-meet/transcripts:Z
    environment:
      - AMPLITUDE_ID
      - ANALYTICS_SCRIPT_URLS
      - ANALYTICS_WHITELISTED_EVENTS
      - AUDIO_QUALITY_OPUS_BITRATE
      - BRANDING_DATA_URL
      - CALLSTATS_CUSTOM_SCRIPT_URL
      - CALLSTATS_ID
      - CALLSTATS_SECRET
      - CHROME_EXTENSION_BANNER_JSON
      - CONFCODE_URL
      - CONFIG_EXTERNAL_CONNECT
      - DEFAULT_LANGUAGE
      - DEPLOYMENTINFO_ENVIRONMENT
      - DEPLOYMENTINFO_ENVIRONMENT_TYPE
      - DEPLOYMENTINFO_REGION
      - DEPLOYMENTINFO_SHARD
      - DEPLOYMENTINFO_USERREGION
```

- DESKTOP\_SHARING\_FRAMERATE\_MIN
- DESKTOP\_SHARING\_FRAMERATE\_MAX
- DIALIN\_NUMBERS\_URL
- DIALOUD\_AUTH\_URL
- DIALOUD\_CODES\_URL
- DISABLE\_AUDIO\_LEVELS
- DISABLE\_DEEP\_LINKING
- DISABLE\_GRANT\_MODERATOR
- DISABLE\_HTTPS
- DISABLE\_KICKOUT
- DISABLE\_LOCAL\_RECORDING
- DISABLE\_POLLS
- DISABLE\_PRIVATE\_CHAT
- DISABLE\_PROFILE
- DISABLE\_REACTIONS
- DISABLE\_REMOTE\_VIDEO\_MENU
- DROPBOX\_APPKEY
- DROPBOX\_REDIRECT\_URI
- DYNAMIC\_BRANDING\_URL
- ENABLE\_AUDIO\_PROCESSING
- ENABLE\_AUTH
- ENABLE\_BREAKOUT\_ROOMS
- ENABLE\_CALENDAR
- ENABLE\_COLIBRI\_WEBSOCKET
- ENABLE\_E2EPING
- ENABLE\_FILE\_RECORDING\_SHARING
- ENABLE\_GUESTS
- ENABLE\_HSTS
- ENABLE\_HTTP\_REDIRECT
- ENABLE\_IPV6
- ENABLE\_LETSENCRYPT
- ENABLE\_LIPSYNC
- ENABLE\_NO\_AUDIO\_DETECTION
- ENABLE\_NOISY\_MIC\_DETECTION
- ENABLE\_OCTO
- ENABLE\_OPUS\_RED
- ENABLE\_PREJOIN\_PAGE
- ENABLE\_P2P
- ENABLE\_WELCOME\_PAGE
- ENABLE\_CLOSE\_PAGE
- ENABLE\_LIVESTREAMING
- ENABLE\_LOCAL\_RECORDING\_NOTIFY\_ALL\_PARTICIPANT
- ENABLE\_LOCAL\_RECORDING\_SELF\_START
- ENABLE\_RECORDING
- ENABLE\_REMB
- ENABLE\_REQUIRE\_DISPLAY\_NAME

- ENABLE\_SERVICE\_RECORDING
- ENABLE\_SIMULCAST
- ENABLE\_STATS\_ID
- ENABLE\_STEREO
- ENABLE\_SUBDOMAINS
- ENABLE\_TALK\_WHILE\_MUTED
- ENABLE\_TCC
- ENABLE\_TRANSCRIPTIONS
- ENABLE\_XMPP\_WEBSOCKET
- ENABLE\_JAAS\_COMPONENTS
- ENABLE\_MULTI\_STREAM
- ETHERPAD\_PUBLIC\_URL
- ETHERPAD\_URL\_BASE
- E2EPING\_NUM\_REQUESTS
- E2EPING\_MAX\_CONFERENCE\_SIZE
- E2EPING\_MAX\_MESSAGE\_PER\_SECOND
- GOOGLE\_ANALYTICS\_ID
- GOOGLE\_API\_APP\_CLIENT\_ID
- HIDE\_PREMEETING\_BUTTONS
- HIDE\_PREJOIN\_DISPLAY\_NAME
- HIDE\_PREJOIN\_EXTRA\_BUTTONS
- INVITE\_SERVICE\_URL
- JICOFO\_AUTH\_USER
- LETSENCRYPT\_DOMAIN
- LETSENCRYPT\_EMAIL
- LETSENCRYPT\_USE\_STAGING
- MATOMO\_ENDPOINT
- MATOMO\_SITE\_ID
- MICROSOFT\_API\_APP\_CLIENT\_ID
- NGINX\_RESOLVER
- NGINX\_WORKER\_PROCESSES
- NGINX\_WORKER\_CONNECTIONS
- PEOPLE\_SEARCH\_URL
- PUBLIC\_URL
- P2P\_PREFERRED\_CODEC
- RESOLUTION
- RESOLUTION\_MIN
- RESOLUTION\_WIDTH
- RESOLUTION\_WIDTH\_MIN
- START\_AUDIO\_MUTED
- START\_AUDIO\_ONLY
- START\_BITRATE
- START\_SILENT
- START\_WITH\_AUDIO\_MUTED
- START\_VIDEO\_MUTED
- START\_WITH\_VIDEO\_MUTED

```
- TESTING_CAP_SCREENSHARE_BITRATE
- TESTING_OCTO_PROBABILITY
- TOKEN_AUTH_URL
- TOOLBAR_BUTTONS
- TZ
- VIDEOQUALITY_BITRATE_H264_LOW
- VIDEOQUALITY_BITRATE_H264_STANDARD
- VIDEOQUALITY_BITRATE_H264_HIGH
- VIDEOQUALITY_BITRATE_VP8_LOW
- VIDEOQUALITY_BITRATE_VP8_STANDARD
- VIDEOQUALITY_BITRATE_VP8_HIGH
- VIDEOQUALITY_BITRATE_VP9_LOW
- VIDEOQUALITY_BITRATE_VP9_STANDARD
- VIDEOQUALITY_BITRATE_VP9_HIGH
- VIDEOQUALITY_ENFORCE_PREFERRED_CODEC
- VIDEOQUALITY_PREFERRED_CODEC
- XMPP_AUTH_DOMAIN
- XMPP_BOSH_URL_BASE
- XMPP_DOMAIN
- XMPP_GUEST_DOMAIN
- XMPP_MUC_DOMAIN
- XMPP_RECORDER_DOMAIN
- XMPP_PORT

networks:
  meet.jitsi:

# XMPP server
prosody:
  image: jitsi/prosody:${JITSI_IMAGE_VERSION:-stable-7882}
  restart: ${RESTART_POLICY:-unless-stopped}
  expose:
    - '${XMPP_PORT:-5222}'
    - '5347'
    - '5280'
  volumes:
    - ${CONFIG}/prosody/config:/config:Z
    - ${CONFIG}/prosody-plugins-custom:/prosody-plugins-
custom:Z
  environment:
    - AUTH_TYPE
    - DISABLE_POLLS
    - ENABLE_AUTH
    - ENABLE_AV_MODERATION
    - ENABLE_BREAKOUT_ROOMS
    - ENABLE_END_CONFERENCE
    - ENABLE_GUESTS
```

- ENABLE\_IPV6
- ENABLE\_LOBBY
- ENABLE\_RECORDING
- ENABLE\_XMPP\_WEBSOCKET
- ENABLE\_JAAS\_COMPONENTS
- GC\_TYPE
- GC\_INC\_TH
- GC\_INC\_SPEED
- GC\_INC\_STEP\_SIZE
- GC\_GEN\_MIN\_TH
- GC\_GEN\_MAX\_TH
- GLOBAL\_CONFIG
- GLOBAL\_MODULES
- JIBRI\_RECORDER\_USER
- JIBRI\_RECORDER\_PASSWORD
- JIBRI\_XMPP\_USER
- JIBRI\_XMPP\_PASSWORD
- JICOFO\_AUTH\_USER
- JICOFO\_AUTH\_PASSWORD
- JICOFO\_COMPONENT\_SECRET
- JIGASI\_XMPP\_USER
- JIGASI\_XMPP\_PASSWORD
- JVB\_AUTH\_USER
- JVB\_AUTH\_PASSWORD
- JWT\_APP\_ID
- JWT\_APP\_SECRET
- JWT\_ACCEPTED\_ISSUERS
- JWT\_ACCEPTED\_AUDIENCES
- JWT\_ASAP\_KEYSERVER
- JWT\_ALLOW\_EMPTY
- JWT\_AUTH\_TYPE
- JWT\_ENABLE\_DOMAIN\_VERIFICATION
- JWT\_TOKEN\_AUTH\_MODULE
- MATRIX\_UVS\_URL
- MATRIX\_UVS\_ISSUER
- MATRIX\_UVS\_AUTH\_TOKEN
- MATRIX\_UVS\_SYNC\_POWER\_LEVELS
- LOG\_LEVEL
- LDAP\_AUTH\_METHOD
- LDAP\_BASE
- LDAP\_BINDDN
- LDAP\_BINDPW
- LDAP\_FILTER
- LDAP\_VERSION
- LDAP\_TLS\_CIPHERS
- LDAP\_TLS\_CHECK\_PEER

```
- LDAP_TLS_CACERT_FILE
- LDAP_TLS_CACERT_DIR
- LDAP_START_TLS
- LDAP_URL
- LDAP_USE_TLS
- MAX_PARTICIPANTS
- PROSODY_RESERVATION_ENABLED
- PROSODY_RESERVATION_REST_BASE_URL
- PUBLIC_URL
- TURN_CREDENTIALS
- TURN_HOST
- TURNS_HOST
- TURN_PORT
- TURNS_PORT
- TZ
- XMPP_DOMAIN
- XMPP_AUTH_DOMAIN
- XMPP_GUEST_DOMAIN
- XMPP_MUC_DOMAIN
- XMPP_INTERNAL_MUC_DOMAIN
- XMPP_MODULES
- XMPP_MUC_MODULES
- XMPP_MUC_CONFIGURATION
- XMPP_INTERNAL_MUC_MODULES
- XMPP_RECORDER_DOMAIN
- XMPP_PORT

networks:
  meet.jitsi:
    aliases:
      - ${XMPP_SERVER:-xmpp.meet.jitsi}

# Focus component
jicofo:
  image: jitsi/jicofo:${JITSI_IMAGE_VERSION:-stable-7882}
  restart: ${RESTART_POLICY:-unless-stopped}
  volumes:
    - ${CONFIG}/jicofo:/config:Z
  environment:
    - AUTH_TYPE
    - BRIDGE_AVG_PARTICIPANT_STRESS
    - BRIDGE_STRESS_THRESHOLD
    - ENABLE_AUTH
    - ENABLE_AUTO_OWNER
    - ENABLE_CODEC_VP8
    - ENABLE_CODEC_VP9
    - ENABLE_CODEC_H264
```

```

        - ENABLE_OCTO
        - ENABLE_RECORDING
        - ENABLE_SCTP
        - ENABLE_AUTO_LOGIN
        - JICOFO_AUTH_USER
        - JICOFO_AUTH_PASSWORD
        - JICOFO_ENABLE_BRIDGE_HEALTH_CHECKS
        - JICOFO_CONF_INITIAL_PARTICIPANT_WAIT_TIMEOUT
        - JICOFO_CONF_SINGLE_PARTICIPANT_TIMEOUT
        - JICOFO_ENABLE_HEALTH_CHECKS
        - JICOFO_SHORT_ID
        - JIBRI_BREWERY_MUC
        - JIBRI_REQUEST_RETRIES
        - JIBRI_PENDING_TIMEOUT
        - JIGASI_BREWERY_MUC
        - JIGASI_SIP_URI
        - JVB_BREWERY_MUC
        - MAX_BRIDGE_PARTICIPANTS
        - OCTO_BRIDGE_SELECTION_STRATEGY
        - SENTRY_DSN="${JICOFO_SENTRY_DSN:-0}"
        - SENTRY_ENVIRONMENT
        - SENTRY_RELEASE
        - TZ
        - XMPP_DOMAIN
        - XMPP_AUTH_DOMAIN
        - XMPP_INTERNAL_MUC_DOMAIN
        - XMPP_MUC_DOMAIN
        - XMPP_RECORDER_DOMAIN
        - XMPP_SERVER
        - XMPP_PORT

depends_on:
    - prosody
networks:
    meet.jitsi:

# Video bridge
jvb:
    image: jitsi/jvb:${JITSI_IMAGE_VERSION:-stable-7882}
    restart: ${RESTART_POLICY:-unless-stopped}
    ports:
        - '127.0.0.1:10000:${JVB_PORT:-10000}/udp'
        - '127.0.0.1:8080:8080'
    volumes:
        - ${CONFIG}/jvb:/config:z
    environment:
        - DOCKER_HOST_ADDRESS

```

```

        - ENABLE_COLIBRI_WEBSOCKET
        - ENABLE_OCTO
        - ENABLE_MULTI_STREAM
        - JVB_ADVERTISE_IPS
        - JVB_ADVERTISE_PRIVATE_CANDIDATES
        - JVB_AUTH_USER
        - JVB_AUTH_PASSWORD
        - JVB_BREWERY_MUC
        - JVB_DISABLE_STUN
        - JVB_PORT
        - JVB_MUC_NICKNAME
        - JVB_STUN_SERVERS
        - JVB_OCTO_BIND_ADDRESS
        - JVB_OCTO_REGION
        - JVB_OCTO_RELAY_ID
        - JVB_WS_DOMAIN
        - JVB_WS_SERVER_ID
        - PUBLIC_URL
        - SENTRY_DSN="${JVB_SENTRY_DSN:-0}"
        - SENTRY_ENVIRONMENT
        - SENTRY_RELEASE
        - COLIBRI_REST_ENABLED
        - SHUTDOWN_REST_ENABLED
        - TZ
        - XMPP_AUTH_DOMAIN
        - XMPP_INTERNAL_MUC_DOMAIN
        - XMPP_SERVER
        - XMPP_PORT

depends_on:
    - prosody
networks:
    meet.jitsi:

# Custom network so all services can communicate using a FQDN
networks:
    meet.jitsi:

```

Después, ejecutamos el script `gen-passwords.sh` para generar las contraseñas en el `.env`:

```
./gen-passwords.sh
```

Tras generar las credenciales, modificamos el archivo `.env` para que quede como el siguiente:

```
# shellcheck disable=SC2034

#####
#####
#####
# Welcome to the Jitsi Meet Docker setup!
#
# This sample .env file contains some basic options to get you started.
# The full options reference can be found here:
# https://jitsi.github.io/handbook/docs/devops-guide/devops-guide-docker
#####
#####
#####

#
# Basic configuration options
#


# Directory where all configuration will be stored
CONFIG=~/.jitsi-meet-cfg

# Exposed HTTP port
#HTTP_PORT=8000

# Exposed HTTPS port
HTTPS_PORT=8445

# System time zone
TZ=UTC

# Public URL for the web service (required)
PUBLIC_URL=https://meet.lab.local

# Media IP addresses to advertise by the JVB
# This setting deprecates DOCKER_HOST_ADDRESS, and supports a comma
separated list of IPs
# See the "Running behind NAT or on a LAN environment" section in the
Handbook:
# https://jitsi.github.io/handbook/docs/devops-guide/devops-guide-
docker#running-behind-nat-or-on-a-lan-environment
#JVB_ADVERTISE_IPS=192.168.1.1,1.2.3.4
```

```
#  
# JaaS Components (beta)  
# https://jaas.8x8.vc  
  
#  
  
# Enable JaaS Components (hosted Jigasi)  
#ENABLE_JAAS_COMPONENTS=0  
  
#  
# Let's Encrypt configuration  
#  
  
# Enable Let's Encrypt certificate generation  
#ENABLE_LETSENCRYPT=1  
  
# Domain for which to generate the certificate  
#LETSENCRYPT_DOMAIN=jitsi.edwo.space  
  
# E-Mail for receiving important account notifications (mandatory)  
#LETSENCRYPT_EMAIL=marc.hortee@gmail.com  
  
# Use the staging server (for avoiding rate limits while testing)  
#LETSENCRYPT_USE_STAGING=1  
  
#  
# Etherpad integration (for document sharing)  
#  
  
# Set etherpad-lite URL in docker local network (uncomment to enable)  
#ETHERPAD_URL_BASE=http://etherpad.meet.jitsi:9001  
  
# Set etherpad-lite public URL, including /p/ pad path fragment (uncomment  
# to enable)  
#ETHERPAD_PUBLIC_URL=https://etherpad.my.domain/p/  
  
# Name your etherpad instance!  
ETHERPAD_TITLE=Video Chat  
  
# The default text of a pad  
ETHERPAD_DEFAULT_PAD_TEXT="Welcome to Web Chat!\n\n"  
  
# Name of the skin for etherpad  
ETHERPAD_SKIN_NAME=colibris  
  
# Skin variants for etherpad
```

```
ETHERPAD_SKIN_VARIANTS="super-light-toolbar super-light-editor light-
background full-width-editor"

#
# Basic Jigasi configuration options (needed for SIP gateway support)
#

# SIP URI for incoming / outgoing calls
#JIGASI_SIP_URI=test@sip2sip.info

# Password for the specified SIP account as a clear text
#JIGASI_SIP_PASSWORD=passw0rd

# SIP server (use the SIP account domain if in doubt)
#JIGASI_SIP_SERVER=sip2sip.info

# SIP server port
#JIGASI_SIP_PORT=5060

# SIP server transport
#JIGASI_SIP_TRANSPORT=UDP


#
# Authentication configuration (see handbook for details)
#

# Enable authentication
#ENABLE_AUTH=1

# Enable guest access
#ENABLE_GUESTS=1

# Select authentication type: internal, jwt, ldap or matrix
#AUTH_TYPE=internal

# JWT authentication
#

# Application identifier
#JWT_APP_ID=my_jitsi_app_id

# Application secret known only to your token generator
#JWT_APP_SECRET=my_jitsi_app_secret
```

```
# (Optional) Set asap_accepted_issuers as a comma separated list
#JWT_ACCEPTED_ISSUERS=my_web_client,my_app_client

# (Optional) Set asap_accepted_audiences as a comma separated list
#JWT_ACCEPTED_AUDIENCES=my_server1,my_server2

# LDAP authentication (for more information see the Cyrus SASL
# saslauthd.conf man page)
#
# LDAP url for connection
#LDAP_URL=ldaps://ldap.domain.com/

# LDAP base DN. Can be empty
#LDAP_BASE=DC=example,DC=domain,DC=com

# LDAP user DN. Do not specify this parameter for the anonymous bind
#LDAP_BINDDN=CN=binduser,OU=users,DC=example,DC=domain,DC=com

# LDAP user password. Do not specify this parameter for the anonymous bind
#LDAP_BINDPW=LdapUserPassw0rd

# LDAP filter. Tokens example:
# %1-9 - if the input key is user@mail.domain.com, then %1 is com, %2 is
# domain and %3 is mail
# %s - %s is replaced by the complete service string
# %r - %r is replaced by the complete realm string
#LDAP_FILTER=(sAMAccountName=%u)

# LDAP authentication method
#LDAP_AUTH_METHOD=bind

# LDAP version
#LDAP_VERSION=3

# LDAP TLS using
#LDAP_USE_TLS=1

# List of SSL/TLS ciphers to allow
#LDAP_TLS_CIPHERS=SECURE256:SECURE128:!AES-128-CBC:!ARCFOUR-128:!CAMELLIA-
#128-CBC:!3DES-CBC:!CAMELLIA-128-CBC

# Require and verify server certificate
#LDAP_TLS_CHECK_PEER=1

# Path to CA cert file. Used when server certificate verify is enabled
```

```
#LDAP_TLS_CACERT_FILE=/etc/ssl/certs/ca-certificates.crt

# Path to CA certs directory. Used when server certificate verify is enabled
#LDAP_TLS_CACERT_DIR=/etc/ssl/certs

# Whether to use starttls, implies LDAPv3 and requires ldap:// instead of
#ldaps://
# LDAP_START_TLS=1


#
# Security
#
# Set these to strong passwords to avoid intruders from impersonating a
service account
# The service(s) won't start unless these are specified
# Running ./gen-passwords.sh will update .env with strong passwords
# You may skip the Jigasi and Jibri passwords if you are not using those
# DO NOT reuse passwords
#


ENABLE_XMPP_WEBSOCKET=0

# XMPP password for Jicofo client connections
JICOFO_AUTH_PASSWORD=bf0f04a34c5ea2b2c785865818c5c18f

# XMPP password for JVB client connections
JVB_AUTH_PASSWORD=729d7b778d4b5f0ba315cf04e8e36e35

# XMPP password for Jigasi MUC client connections
JIGASI_XMPP_PASSWORD=dd815df08f4f95af21af8ff94eeceeee

# XMPP recorder password for Jibri client connections
JIBRI_RECORDER_PASSWORD=9bcd5ed78e2c9fc2b8e1c02776f98aff

# XMPP password for Jibri client connections
JIBRI_XMPP_PASSWORD=015c2314a37ba51d53eb3da9695a22fc


#
# Docker Compose options
#


# Container restart policy
#RESTART_POLICY=unless-stopped
```

```
# Jitsi image version (useful for local development)
#JITSI_IMAGE_VERSION=latest
```

Basicamente hemos cambiado:

- **HTTPS\_PORT** a `8445` (para bindear el servicio web a ese puerto).
- **PUBLIC\_URL** a `https://meet.lab.local` (la URL que utilizaremos para visitar el jitsi).

#### Note

Se ha deshabilitado el uso de Websockets con el parámetro:

`ENABLE_XMPP_WEBSOCKET=0`, debido a que, al utilizar un proxy inverso como Nginx, no ha sido posible hacer pasar los Websockets a través del Reverse Proxy.

Con esto listo, procedemos a realizar el despliegue con:

```
docker compose up -d
```

Tras el despliegue podemos ver como el contenedor esta corriendo con:

```
docker compose ps
```

```
root@vps:/srv/containers/jitsi# docker compose ps
NAME      COMMAND     SERVICE      STATUS      PORTS
jitsi-jicofo-1    "/init"     jicofo      running
jitsi-jvb-1       "/init"     jvb        running
jitsi-prosody-1   "/init"     prosody     running
jitsi-web-1       "/init"     web        running
root@vps:/srv/containers/jitsi#
```

Se añade el subdominio al `/etc/hosts` para gestionar el servicio (como se ha hecho anteriormente) quedando el archivo así:

```
127.0.0.1 localhost
127.0.1.1 vps

# The following lines are desirable for IPv6 capable hosts
::1      ip6-localhost ip6-loopback
fe00::0  ip6-localnet
ff00::0  ip6-mcastprefix
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters

# LAB.LOCAL DNS records
192.168.10.1 vps.lab.local
```

```
192.168.10.2 laptop.lab.local
```

```
192.168.10.1 git.lab.local  
192.168.10.1 md.lab.local  
192.168.10.1 chat.lab.local  
192.168.10.1 pass.lab.local  
192.168.10.1 wekan.lab.local  
192.168.10.1 wiki.lab.local  
192.168.10.1 meet.lab.local
```

También se crea el archivo de configuración de Nginx para gestionar el virtual host. Este archivo se sitúa en `/etc/nginx/sites-available/meet.lab.local` y tiene el siguiente contenido:

```
server {  
    listen 192.168.10.1:80;  
    server_name meet.lab.local;  
    # Redirect all non-https requests  
    rewrite ^ https://$host$request_uri? permanent;  
}  
  
server {  
    listen 192.168.10.1:443 ssl http2;  
  
    include snippets/self-signed.conf;  
    include snippets/ssl-params.conf;  
  
    server_name meet.lab.local;  
  
    client_max_body_size 0;  
    proxy_http_version 1.1;  
    proxy_request_buffering off;  
  
    location / {  
        proxy_pass https://127.0.0.1:8445;  
        proxy_set_header Host $host;  
        proxy_set_header X-Real-IP $remote_addr;  
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
        proxy_set_header X-Forwarded-Proto $scheme;  
    }  
}
```

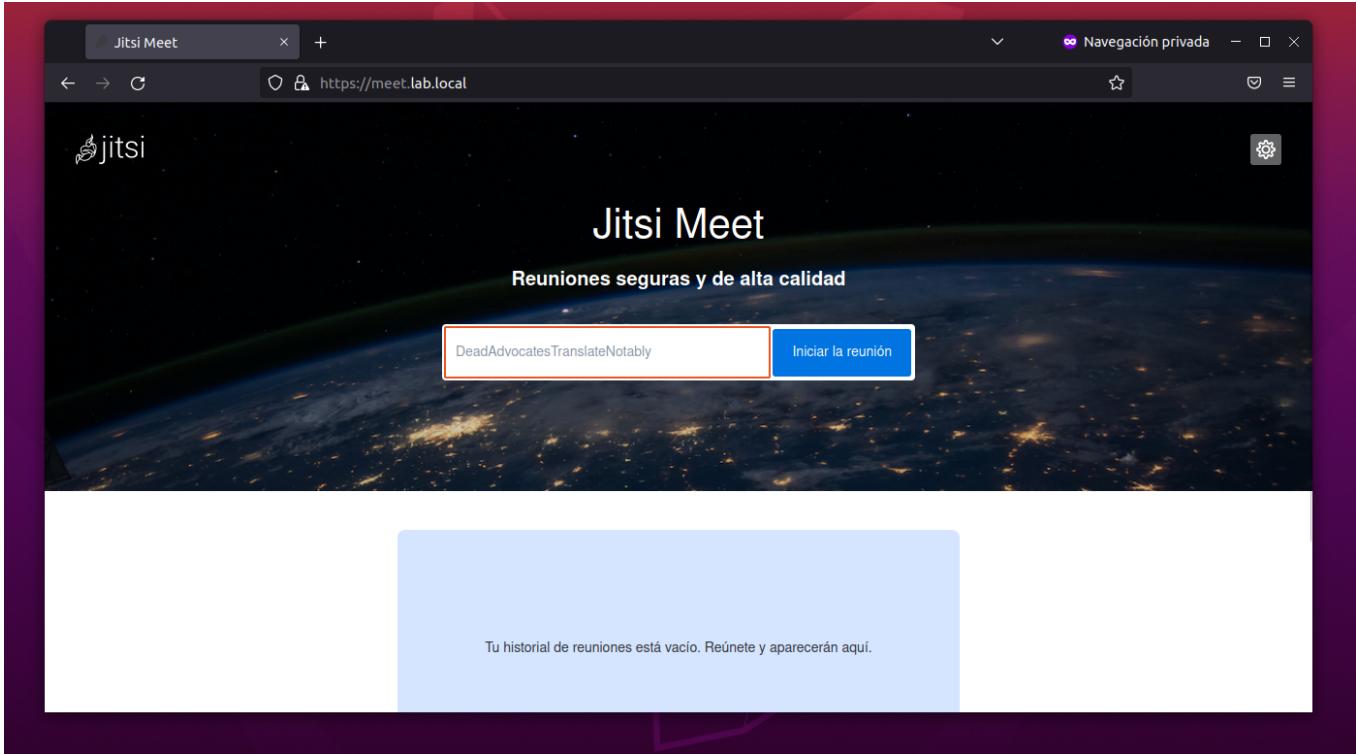
Tras esto, se crea el enlace simbólico para activarlo:

```
ln -s /etc/nginx/sites-available/meet.lab.local /etc/nginx/sites-enabled/meet.lab.local
```

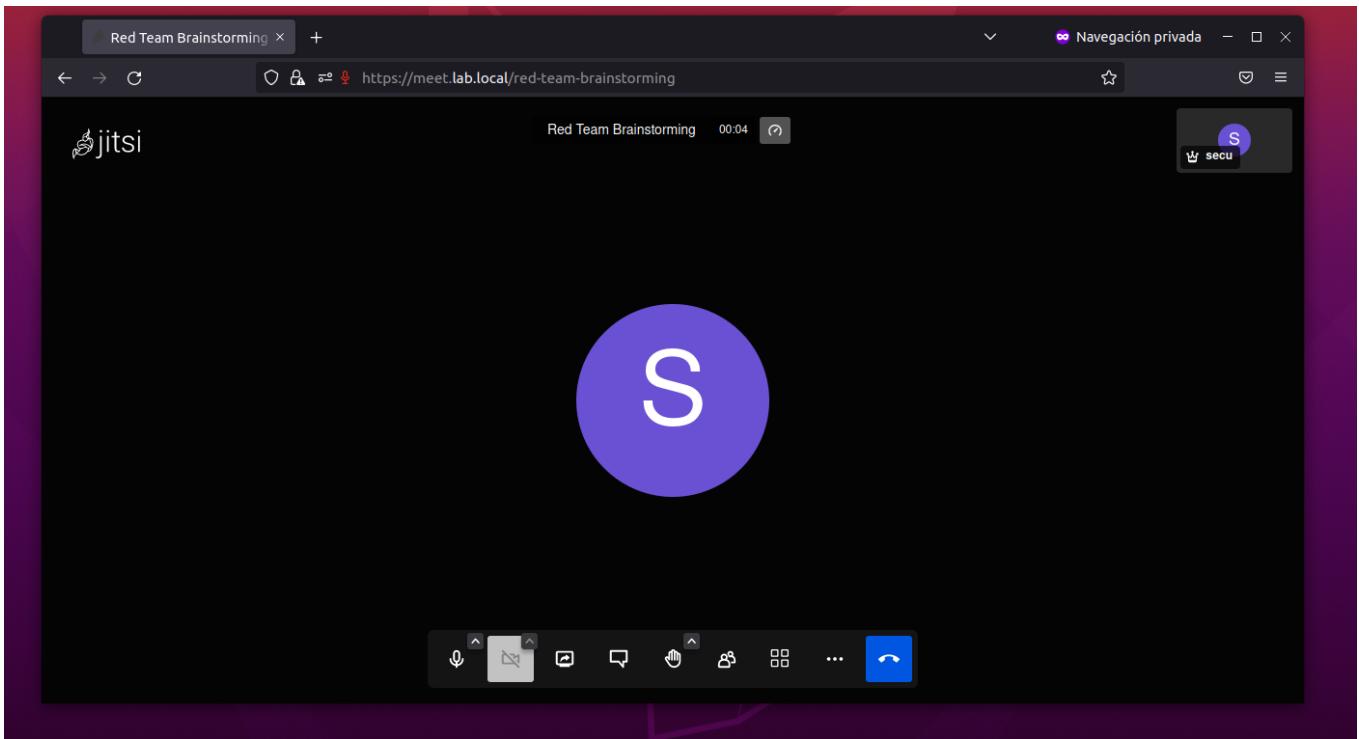
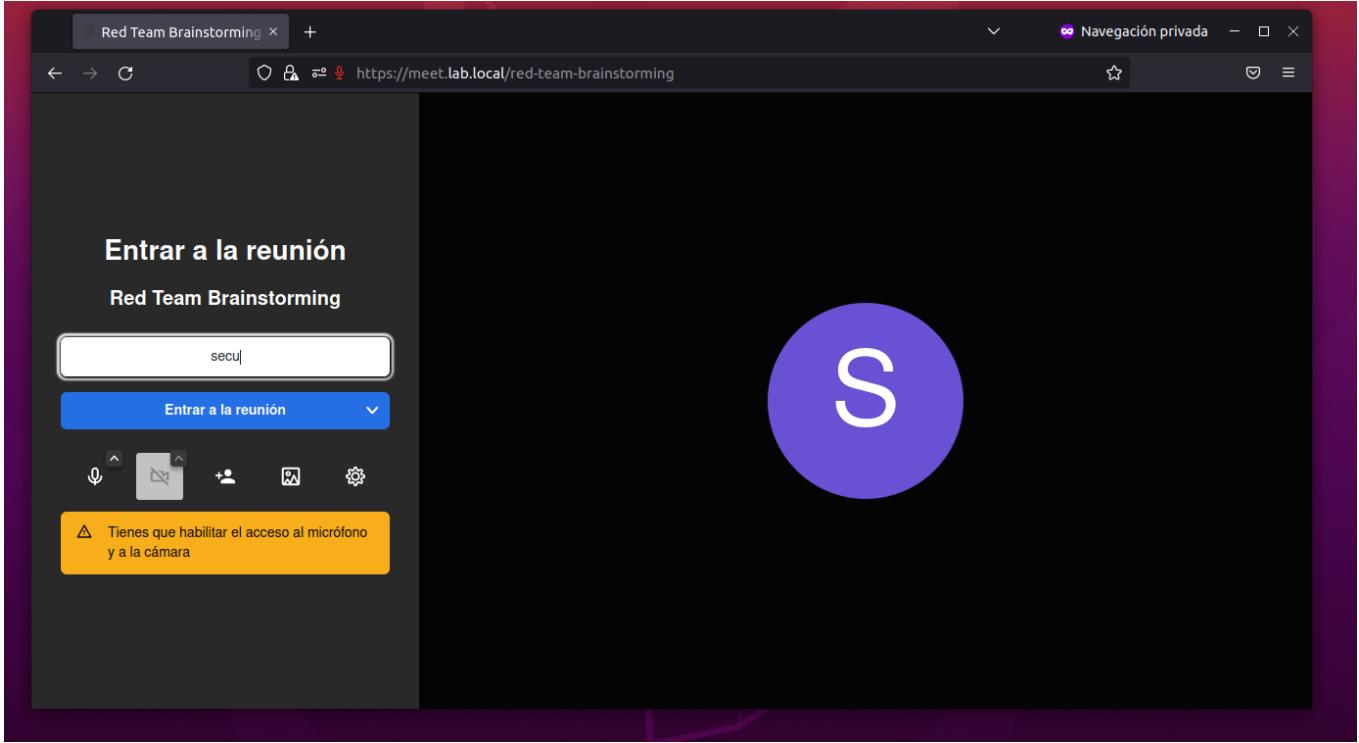
Se comprueba que todo esta correcto y se reinicia el servicio de Nginx y Dnsmasq para aplicar los cambios:

```
nginx -t  
systemctl restart nginx  
systemctl restart dnsmasq
```

Y, desde la máquina **LAPTOP**, visitamos la dirección <https://meet.lab.local> (tras excepcionar el certificado SSL) encontramos:



En esta guía no entraremos a indagar en las funcionalidades de Jitsi, simplemente mostramos que se puede acceder a una sala y funciona correctamente:



Y llegados a este punto, ya disponemos de todos los servicios dockerizados en la Infraestructura. En los siguientes pasos continuaremos con la integración de una máquina física (Virtualizador) y como se puede acceder a los servicios de esta, utilizando la enrutación de paquetes a través de la VPN.

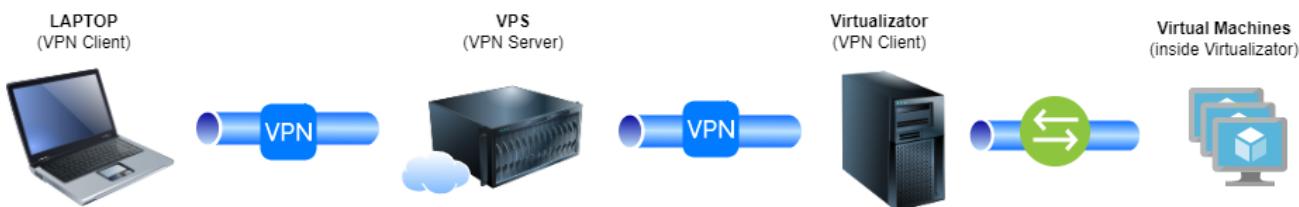
# Integración con Virtualizador Físico

En este punto de la guía, disponemos de los servicios de la Infraestructura desplegados y funcionando. El siguiente paso que se va a realizar es integrar lo que llamamos "Virtualizador Físico".

Un **Virtualizador Físico** no es otra cosa que una equipo físico (es decir un portatil u ordenador de sobremesa) donde se instalará un servicio de virtualización, como lo es VirtualBox, VMware o KVM, y lo conectaremos a la VPN.

La idea es que, cualquier miembro de la organización, pueda acceder a las máquinas virtuales contenidas en el equipo físico que actúa como cliente de la VPN.

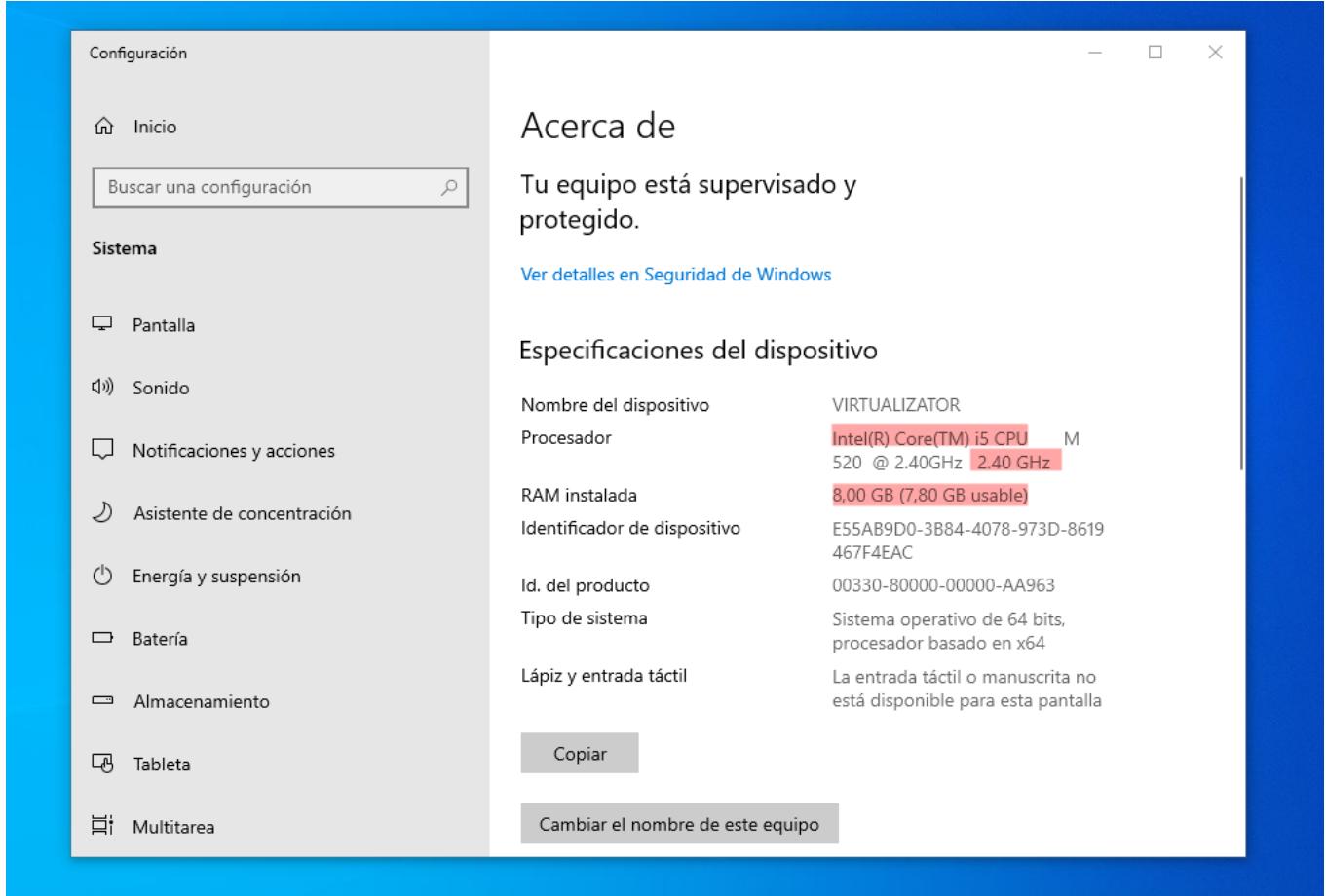
Una esquema visual del proceso de comunicación entre los clientes VPN y las máquinas virtuales sería el siguiente:



La máquina física que vamos a utilizar como "Virtualizador" para esta guía, se trata de un equipo Windows con un procesador Intel Core i5, 8 Gb de RAM y 300GB de espacio en disco.

## Note

La idea de utilizar este equipo, es mostrar que, en ocasiones, puede ser interesante reutilizar un equipo antiguo (que ya no se usa), y reacondicionarlo para alojar máquinas virtuales u otros servicios y poder acceder a ellos a través de la VPN.



## Despliegue de máquina virtual y configuración de red

Para comenzar, dentro del Virtualizador, comenzamos instalando las siguientes aplicaciones:

- [Wireguard](#): se puede encontrar el instalador (exe) en la página oficial.
- [Virtual Box](#): igualmente, se descarga el ejecutable desde la página oficial.

Ambos programas son muy sencillos de instalar en sistemas Windows, por tanto, nos saltaremos este proceso y continuamos con la configuración de red.

El primer paso consiste en generar el perfil de conexión a la VPN del Virtualizador. Este paso es el mismo a cuando generamos los perfiles de conexión para la máquina virtual LAPTOP.

Primero creamos el directorio del nuevo peer en el servidor VPS:

```
mkdir -p /etc/wireguard/peers/virtualizer
```

Después, nos movemos al directorio y generamos el par de claves:

```
cd /etc/wireguard/peers/virtualizer  
wg genkey | tee private.key | wg pubkey > public.key
```

Tras generar las claves las trasladamos a un archivo de configuración del cliente en `/etc/wireguard/peers/virtualizer/wg0.conf` que quedará de la siguiente forma:

```
[Interface]  
Address = 192.168.10.4/32  
PrivateKey = YOGe82HkfC6gm8Gvl4unX8oQijFDLq9PeFCsqkuD0kY=  
DNS = 192.168.10.1  
  
[Peer]  
PublicKey = 6qazAiPNifKVY4EilSkNSbKw4oM8xUclDTsNBX+JVk4=  
Endpoint = 165.22.71.211:41194  
AllowedIPs = 192.168.10.0/24  
PersistentKeepalive = 25
```

### Note

**IMPORTANTE:** ahora podemos ver que el "Endpoint" ha cambiado, pues figura la dirección IP del servidor en la nube (pues, como hemos comentado al principio, este proceso se debe realizar con la instalación realizada en la nube).

Damos de alta al nuevo peer en el servidor de Wireguard editando el archivo `/etc/wireguard/wg0.conf` que quedará como el siguiente:

```
[Interface]  
Address = 192.168.10.1/24  
PrivateKey = KENQSQNZboacb5U34BwtMTqMLg1dfFNIjrUL71TF9mY=  
ListenPort = 41194  
PostUp = iptables -A FORWARD -i %i -j ACCEPT;iptables -A FORWARD -o %i -j  
ACCEPT; iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE  
PostDown = iptables -D FORWARD -i %i -j ACCEPT;iptables -D FORWARD -o %i -j  
ACCEPT; iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE  
  
# Client - LAPTOP  
[Peer]  
PublicKey = 9pFuhQ9s7WN9I3icj5JFRZwmKB3pf03qX+nMz2IfMRU=  
AllowedIPs = 192.168.10.2/32  
  
# Client - PHONE  
[Peer]  
PublicKey = Hz6fEvZhZxkdzzRb05oRRvdasapDXva+pREdrU0atxc=
```

```
AllowedIPs = 192.168.10.3/32

# Client - VIRTUALIZATOR
[Peer]
PublicKey = M1/ZwmXLODdaLnYx622Lwq4ne8p6nPetviPrpijHVRE=
AllowedIPs = 192.168.10.4/32
```

### ⚠ Warning

Aquí hay que actualizar el nuevo nombre de la interfaz de red que figura en el PostUp y PostDown de la configuración. Se debe cambiar al nombre de la interfaz de red que esta conectada a internet.

Finalmente, reiniciamos el servicio de Wireguard con:

```
systemctl restart wg-quick@wg0
```

Y verificamos que el nuevo peer se ha registrado correctamente:

```
root@vps-cloud-server:~# wg
interface: wg0
  public key: 6qazAiPNifKVY4EilSkNSbKw4oM8xUc1DTsNBX+JVk4=
  private key: (hidden)
  listening port: 41194

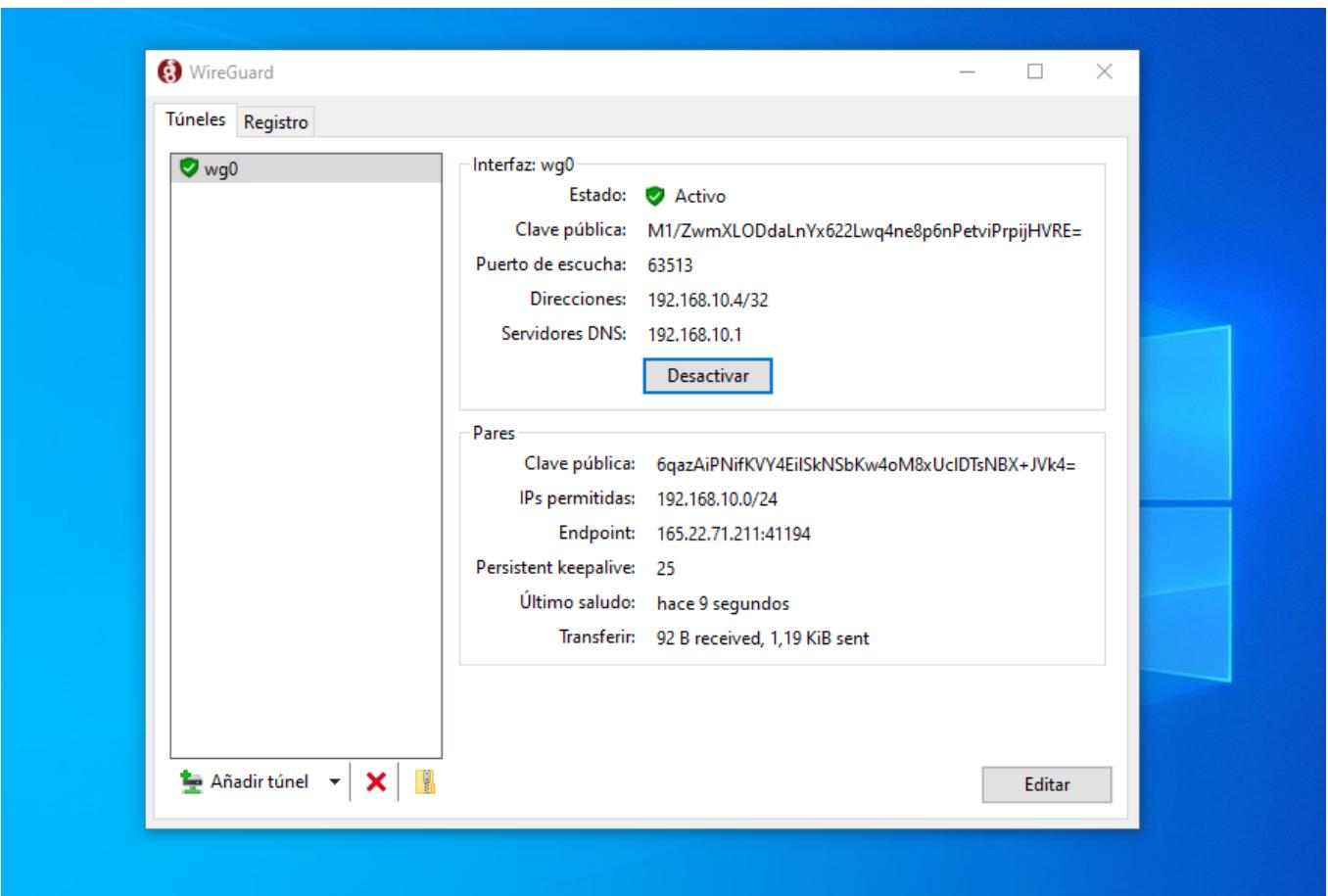
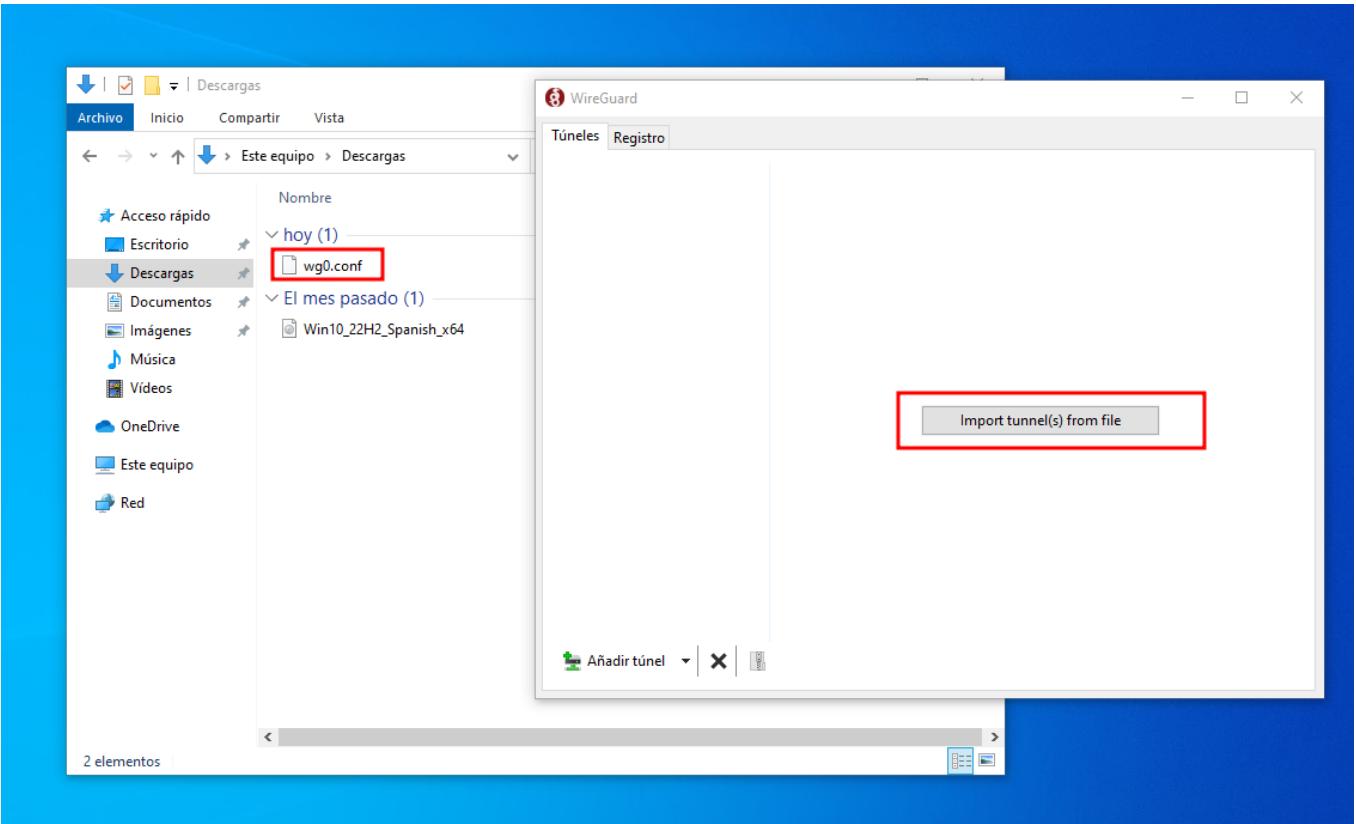
peer: 9pFuhQ9s7WN9I3icj5JFRZwmKB3pf03qX+nMz2IfMRU=
  allowed ips: 192.168.10.2/32

peer: Hz6fEvZhZxkdzzRbO5oRRvdasapDXva+pREdrU0atxc=
  allowed ips: 192.168.10.3/32

peer: M1/ZwmXLODdaLnYx622Lwq4ne8p6nPetviPrpijHVRE=
  allowed ips: 192.168.10.4/32
root@vps-cloud-server:~#
```

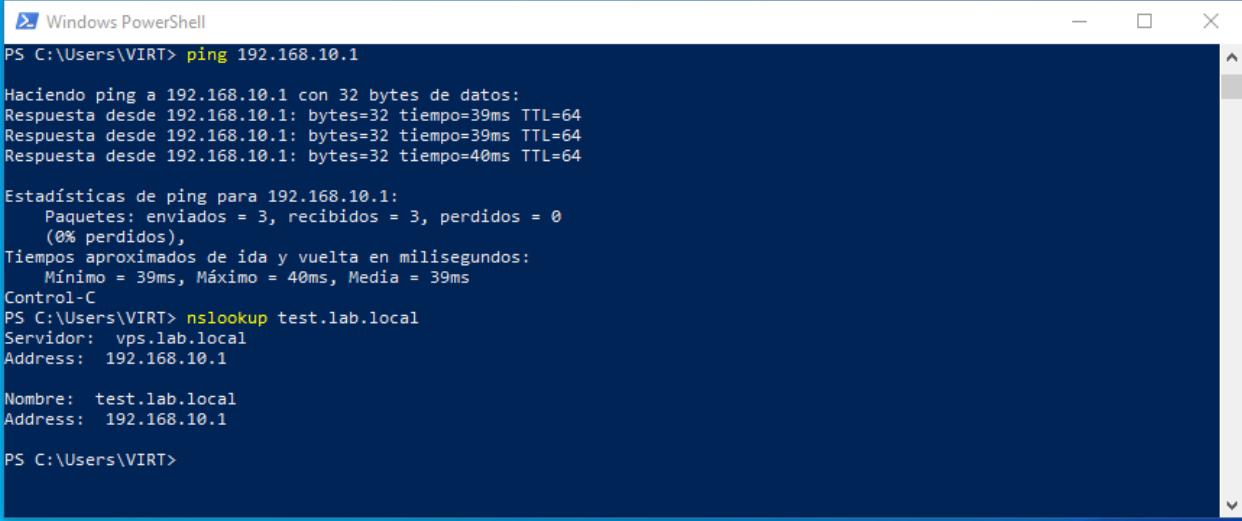
Tras esto, tenemos que copiar el archivo de configuración situado en `/etc/wireguard/peers/virtualizator/wg0.conf` al propio Virtualizador. Puedes hacerlo via SSH, a través de un servidor web o como mejor te venga.

Una vez se tiene el archivo de configuración en la máquina, se procede a importarlo desde la aplicación Windows de Wireguard:

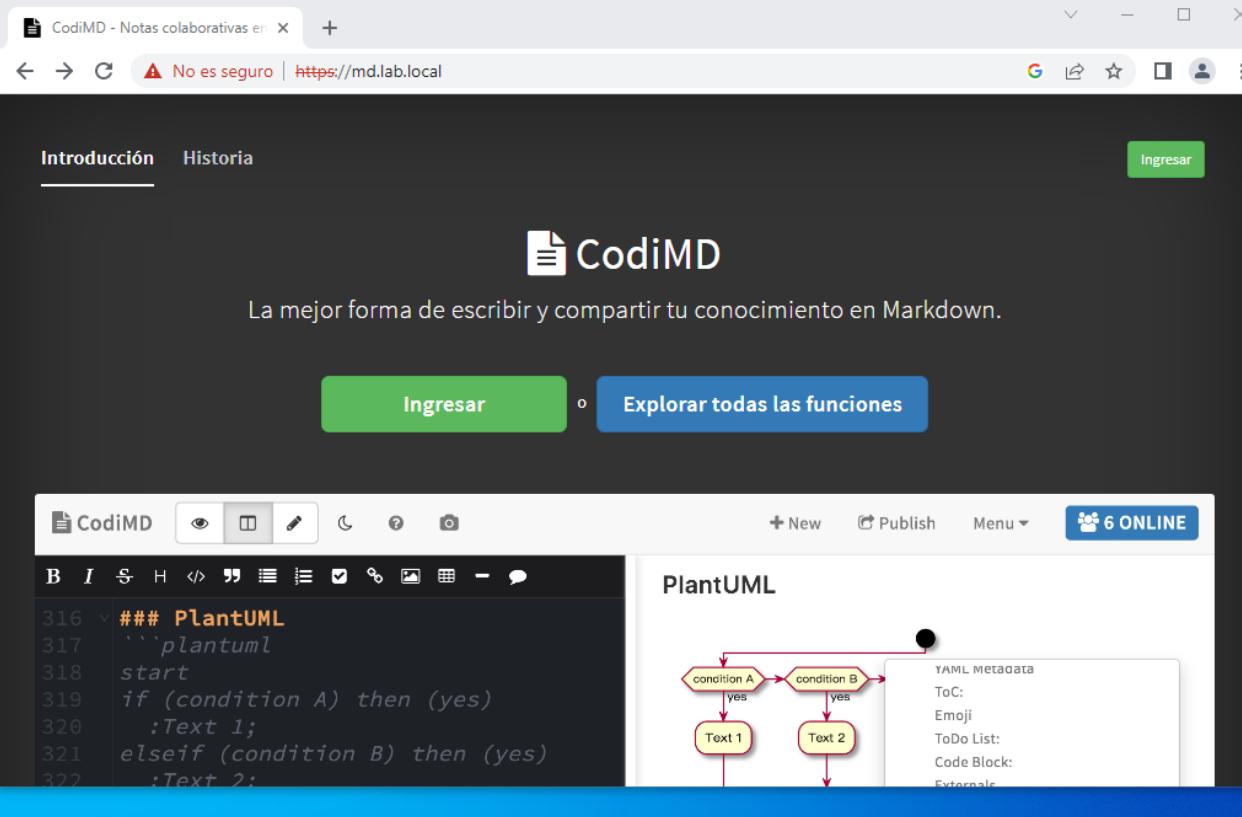


En este momento, comprobamos que nos hemos conectado a la VPN de forma exitosa:

```
ping 192.168.10.1  
nslookup test.lab.local
```

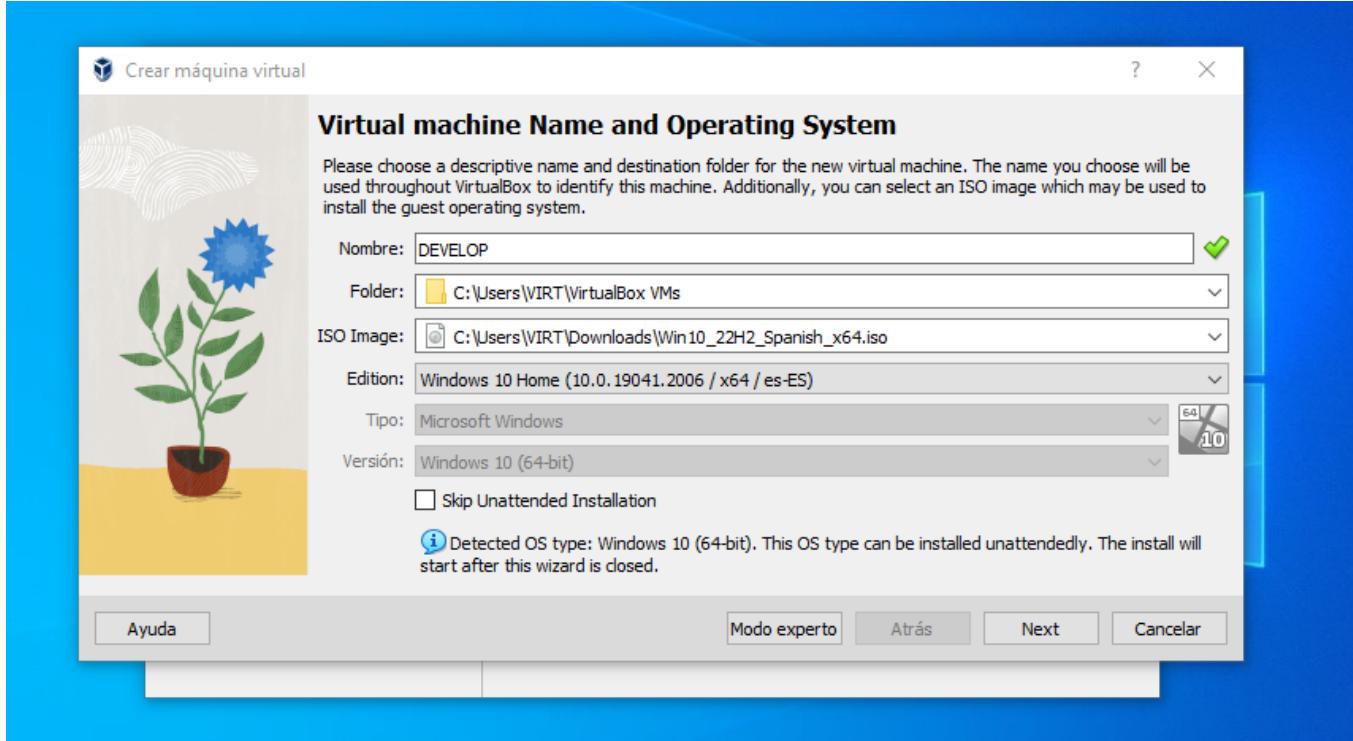


```
Windows PowerShell  
PS C:\Users\VIRT> ping 192.168.10.1  
  
Haciendo ping a 192.168.10.1 con 32 bytes de datos:  
Respuesta desde 192.168.10.1: bytes=32 tiempo=39ms TTL=64  
Respuesta desde 192.168.10.1: bytes=32 tiempo=39ms TTL=64  
Respuesta desde 192.168.10.1: bytes=32 tiempo=40ms TTL=64  
  
Estadísticas de ping para 192.168.10.1:  
Paquetes: enviados = 3, recibidos = 3, perdidos = 0  
(0% perdidos),  
Tiempos aproximados de ida y vuelta en milisegundos:  
Mínimo = 39ms, Máximo = 40ms, Media = 39ms  
Control+C  
PS C:\Users\VIRT> nslookup test.lab.local  
Servidor: vps.lab.local  
Address: 192.168.10.1  
  
Nombre: test.lab.local  
Address: 192.168.10.1  
  
PS C:\Users\VIRT>
```



The screenshot shows the CodiMD web application. At the top, there's a navigation bar with links for 'Introducción' (selected), 'Historia', and 'Ingresar'. Below the navigation is a logo with the text 'CodiMD' and a subtitle 'La mejor forma de escribir y compartir tu conocimiento en Markdown.' There are two buttons: 'Ingresar' (green) and 'Explorar todas las funciones' (blue). The main content area has a dark background. On the left, there's a code editor with syntax highlighting for Markdown, showing code like `316 v ### PlantUML` and `317 ``'plantuml` followed by UML code. On the right, there's a 'PlantUML' diagram editor showing a sequence diagram with nodes 'Text 1' and 'Text 2' connected by arrows labeled 'condition A' and 'condition B'. A sidebar titled 'YAML Metadata' lists features: ToC, Emoji, ToDo List, Code Block, and External. At the bottom right, a blue button says '6 ONLINE'.

Finalmente procedemos a crear una máquina virtual desde el Virtualizador, para este ejemplo yo crearé una máquina virtual Windows 10 Pro.



### Note

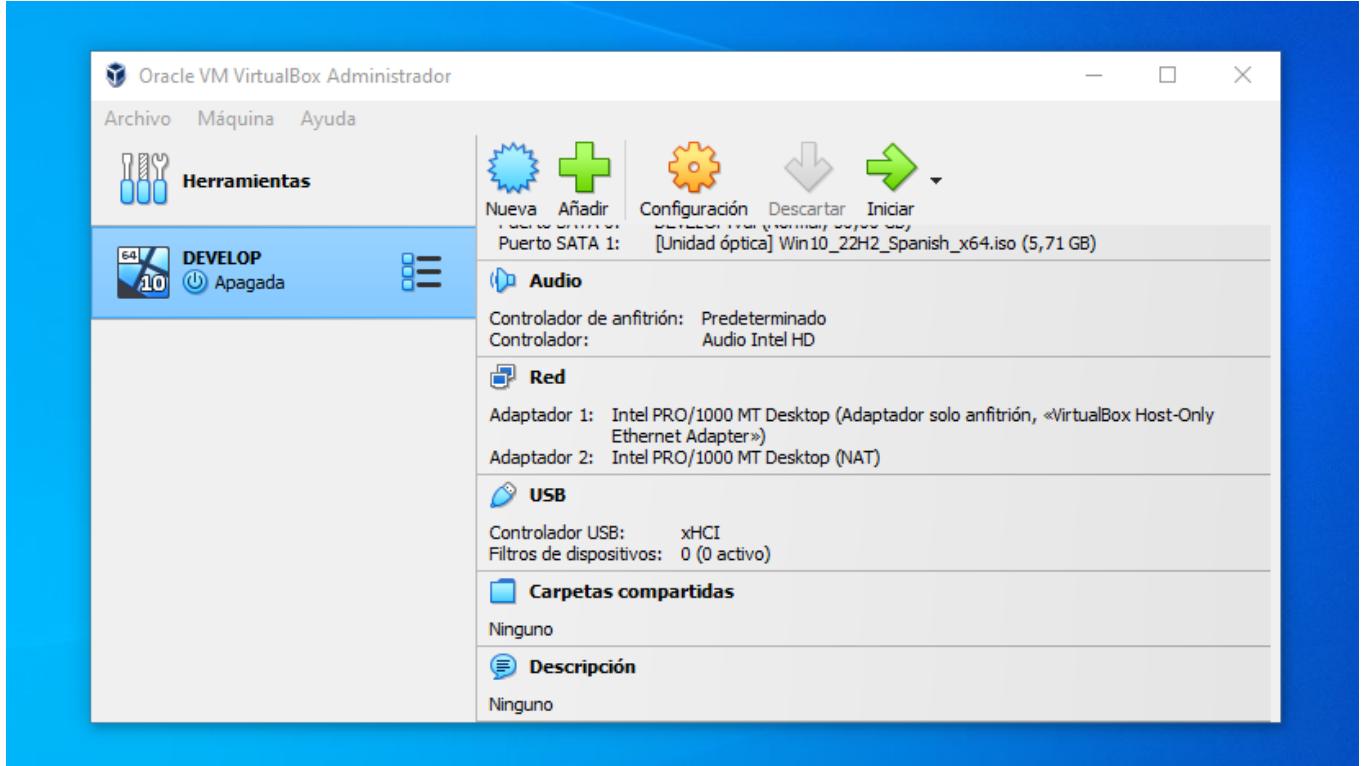
Como sugerencia para un equipo de Red Team, esta máquina virtual, podría ser una máquina que dispone de todas las herramientas de desarrollo y compilación de aplicaciones en C, C++, C#, etc; así como por ejemplo incorporar también utilidades de ingeniería inversa u otra índole.

La creación de la máquina virtual será un paso que nos saltaremos en la guía, ahora bien, algo que si se va a destacar (y que es de vital importancia), es la configuración de red de la máquina virtual. Pues, para que sea accesible desde la VPN, se necesitan realizar una serie de configuraciones.

Debemos configurar en la máquina virtual Windows, 2 adaptadores de red como mínimo:

- 1. Un adaptador sólo-anfitrión:** este adaptador será el que utilicemos para comunicarnos con la máquina host (Virtualizador) y viceversa. Sin esta configuración no podremos hacer el enrutado de información a través de la interfaz de red y llegar a los servicios de la VM.
- 2. Un adaptador de red en modo NAT:** esta configuración es opcional, pues dependerá si quieres que tu máquina este conectada a Internet o no. En mi caso, si configuraré la interfaz para poder realizar las instalaciones de paquetes que sean necesarios.

Tras finalizar la configuración de red, debería quedar de la siguiente forma:

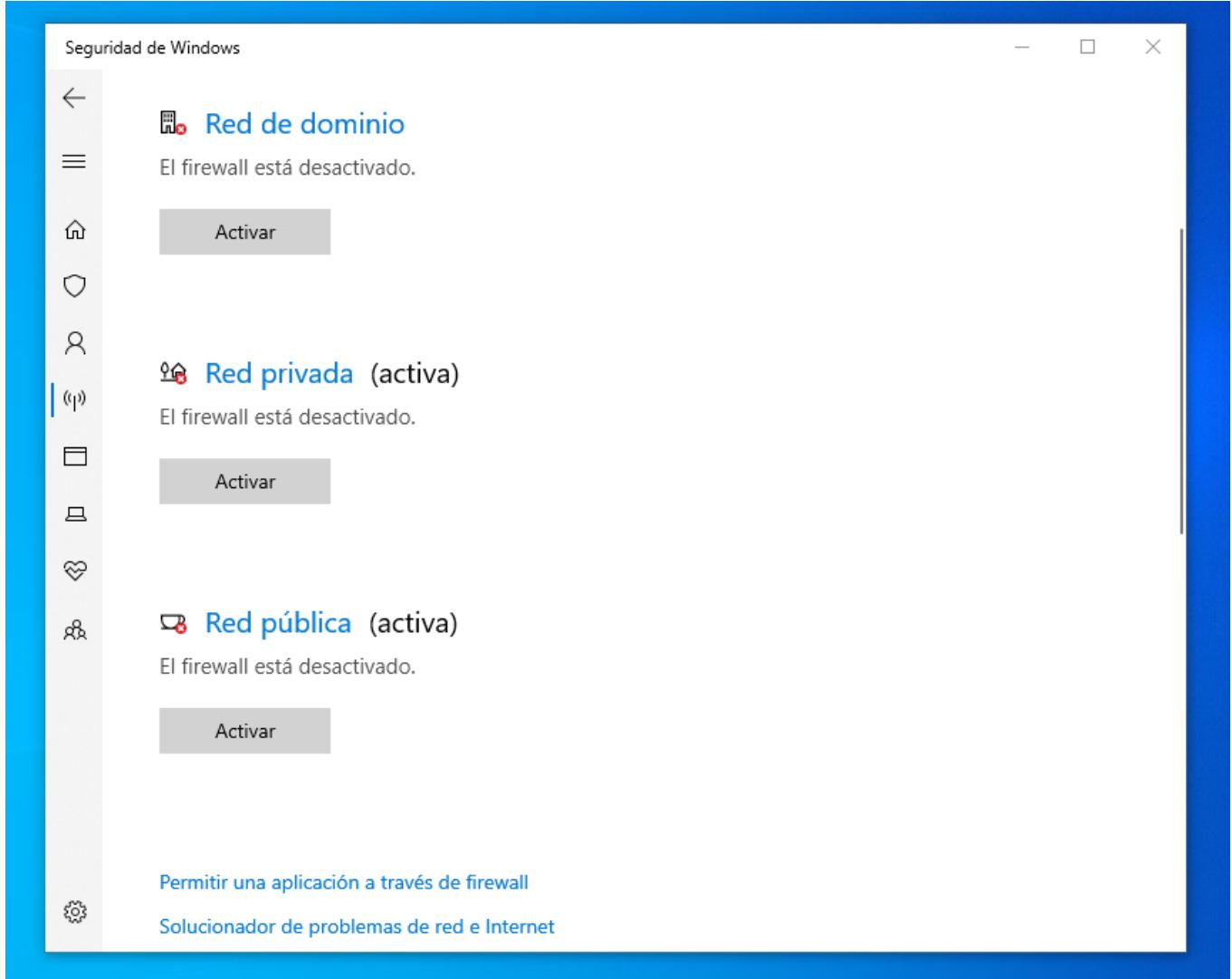


En este momento procedemos con la instalación de la máquina virtual Windows. Este paso se tiene que completar para poder seguir con la guía.

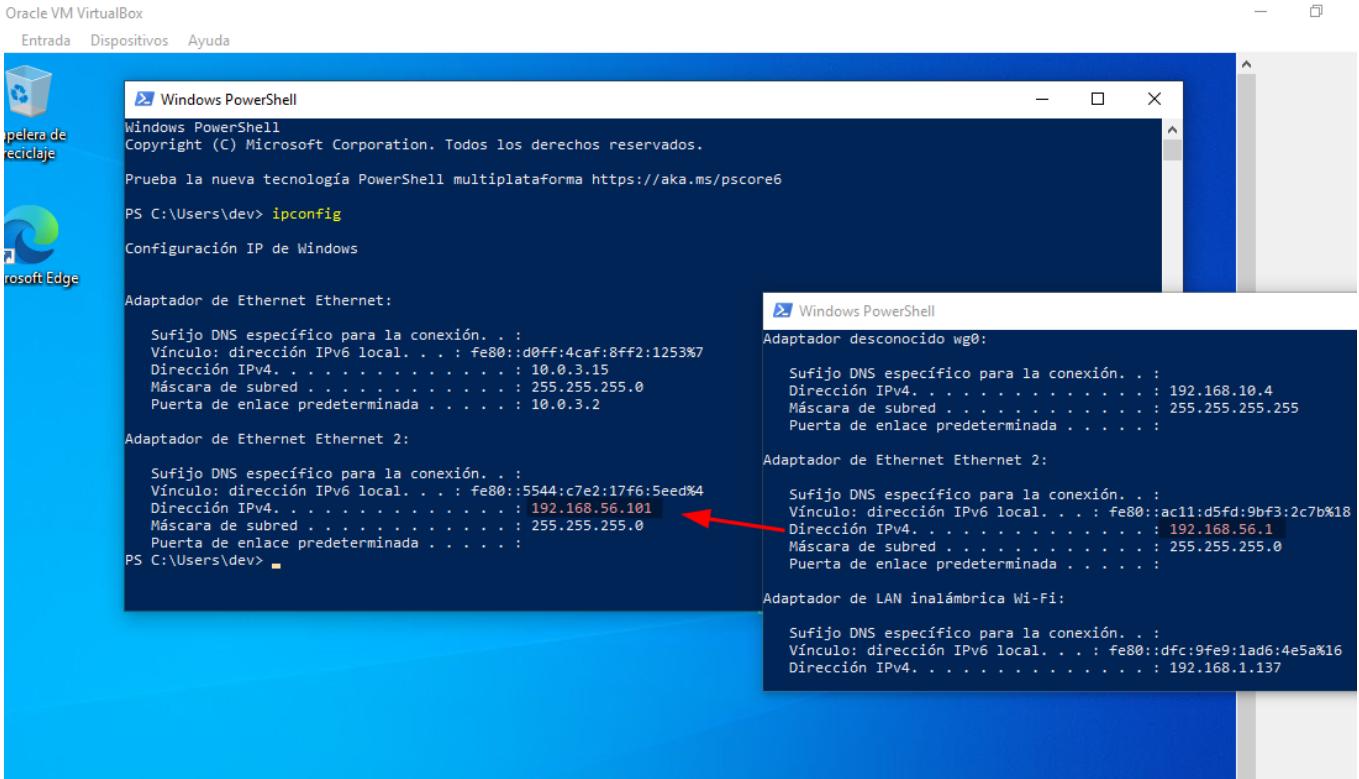
## Creación de reglas de enrutado entre VPN, Virtualizador y VM

Una vez disponemos de la máquina virtual instalada y encendida, procedemos con la configuración de red relacionada con el enruteamiento de paquetes a través de la VPN.

- Primero, aseguramos que el Firewall de Windows esta deshabilitado (o en caso de no querer deshabilitarlo, configurar las reglas necesarias para que se pueda acceder a los servicios de las máquinas). Comprobamos en Virtualizador y Máquina Virtual de Windows.



- Segundo, identificamos el segmento de red del adaptador sólo-anfitrión (que comparte máquina virtual y virtualizador). En mi caso es:
  - **Virtualizador:** 192.168.56.1
  - **Máquina Virtual (develop):** 192.168.56.101



- Tercero, sabiendo el rango de la red que conecta el virtualizador (Cliente VPN) con las máquinas virtuales, se procede a establecer este rango en los archivos de configuración de Wireguard tanto del VPS en la nube (Servidor VPN) como de los Clientes VPN que se vayan a conectar a las máquinas (en este caso LAPTOP).

```
root@vps-cloud-server:/etc/wireguard# cat wg0.conf
[Interface]
Address = 192.168.10.1/24
PrivateKey = KEN0SQNZboacb5U34BwtMTqMLg1dfFNIjrzUL71TF9mY=
ListenPort = 41194
PostUp = iptables -A FORWARD -i %i -j ACCEPT;iptables -A FORWARD -o %i -j ACCEPT; iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT;iptables -D FORWARD -o %i -j ACCEPT; iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE

# Client - LAPTOP
[Peer]
PublicKey = 9pFuhQ9s7WN9I3icj5JFRZwmKB3pf03qX+nMz2IfMRU=
AllowedIPs = 192.168.10.2/32

# Client - PHONE
[Peer]
PublicKey = Hz6fEvZhZxdzzRbO5oRRvdasapDXva+pREdrU0atxc=
AllowedIPs = 192.168.10.3/32

# Client - VIRTUALIZATOR
[Peer]
PublicKey = M1/ZwmXL0DdaLnYx622Lwq4ne8p6nPetviPrpijHVRE=
AllowedIPs = 192.168.10.4/32,192.168.56.0/24
root@vps-cloud-server:/etc/wireguard#
```

```
root@laptop:/etc/wireguard# cat wg0.conf
[Interface]
Address = 192.168.10.2/32
PrivateKey = wE9QH3EZ1bNs4Q1Elh6sW2IEBLQLdcKS077uv2Qv82M=
DNS = 192.168.10.1

[Peer]
PublicKey = 6qazAiPNifKVY4EilSkNSbKw4oM8xUclDTsNBX+JVk4=
Endpoint = 165.22.71.211:41194
AllowedIPs = 192.168.10.0/24, 192.168.56.0/24
PersistentKeepalive = 25
root@laptop:/etc/wireguard#
```

### Note

De esta forma, le indicamos a Wireguard, que debe enrutar los paquetes que vayan a la dirección 192.168.56.0/24 por la interfaz de la VPN. Y luego, en el servidor VPN, le indicamos que ese segmento de red esta gestionado por el cliente vpn:

**VIRTUALIZATOR.**

- Reiniciamos los servicios de Wireguard y comprobamos la tabla de rutas en ambas máquinas para confirmar que se ha definido correctamente.

```
root@vps-cloud-server:/etc/wireguard# route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref  Use Iface
0.0.0.0         165.22.64.1   0.0.0.0        UG    0      0        0 eth0
10.19.0.0       0.0.0.0       255.255.0.0   U     0      0        0 eth0
10.114.0.0      0.0.0.0       255.255.240.0 U     0      0        0 eth1
165.22.64.0     0.0.0.0       255.255.240.0 U     0      0        0 eth0
172.17.0.0      0.0.0.0       255.255.0.0   U     0      0        0 docker0
172.18.0.0      0.0.0.0       255.255.0.0   U     0      0        0 br_hackmd
192.168.10.0    0.0.0.0       255.255.255.0  U     0      0        0 wg0
192.168.56.0    0.0.0.0       255.255.255.0  U     0      0        0 wg0
root@vps-cloud-server:/etc/wireguard#
```

```
root@laptop:/etc/wireguard# route -n
Tabla de rutas IP del núcleo
Destino      Pasarela      Genmask     Indic Métric Ref    Uso Interfaz
0.0.0.0      192.168.145.2 0.0.0.0     UG   102      0        0 ens36
0.0.0.0      192.168.145.2 0.0.0.0     UG   103      0        0 ens33
0.0.0.0      10.10.10.1    0.0.0.0     UG 20101     0        0 ens34
10.10.10.0   0.0.0.0      255.255.255.0 U   101      0        0 ens34
169.254.0.0  0.0.0.0      255.255.0.0  U   1000     0        0 ens34
192.168.10.0 0.0.0.0     255.255.255.0 U   0        0        0 wg0
192.168.56.0 0.0.0.0     255.255.255.0 U   0        0        0 wg0
192.168.145.0 0.0.0.0    255.255.255.0 U   102      0        0 ens36
192.168.145.0 0.0.0.0    255.255.255.0 U   103      0        0 ens33
root@laptop:/etc/wireguard#
```

- El cuarto paso es habilitar el IPv4 Forward tanto en el VPS en la nube (Servidor VPN) como en el Virtualizador (Cliente VPN que enruta):

En el VPS en la nube ejecutamos el comando:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

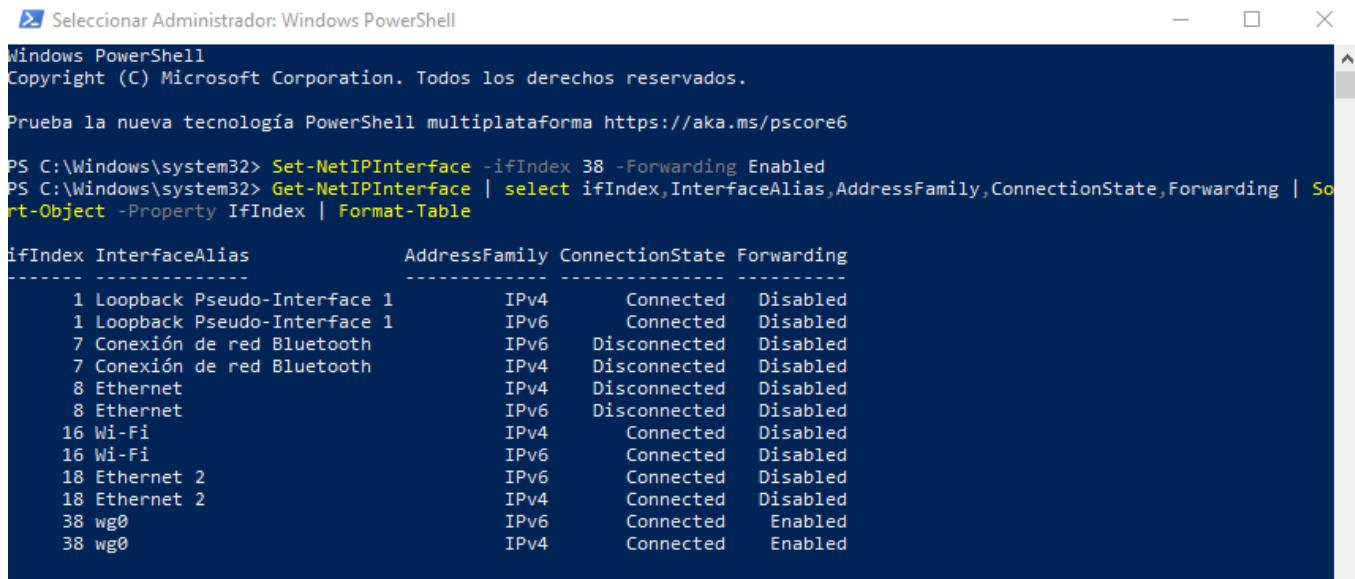
En Virtualizador, desde una consola administrativa de Powershell listamos las interfaces con:

```
Get-NetIpInterface | Select
ifIndex,InterfaceAlias,AddressFamily,ConnectionState,Forwarding | Sort-
Object -Property IfIndex | Format-Table
```

| IfIndex | InterfaceAlias              | AddressFamily | ConnectionState | Forwarding |
|---------|-----------------------------|---------------|-----------------|------------|
| 1       | Loopback Pseudo-Interface 1 | IPv4          | Connected       | Disabled   |
| 1       | Loopback Pseudo-Interface 1 | IPv6          | Connected       | Disabled   |
| 7       | Conección de red Bluetooth  | IPv6          | Disconnected    | Disabled   |
| 7       | Conección de red Bluetooth  | IPv4          | Disconnected    | Disabled   |
| 8       | Ethernet                    | IPv4          | Disconnected    | Disabled   |
| 8       | Ethernet                    | IPv6          | Disconnected    | Disabled   |
| 16      | Wi-Fi                       | IPv4          | Connected       | Disabled   |
| 16      | Wi-Fi                       | IPv6          | Connected       | Disabled   |
| 18      | Ethernet 2                  | IPv6          | Connected       | Disabled   |
| 18      | Ethernet 2                  | IPv4          | Connected       | Disabled   |
| 38      | wg0                         | IPv6          | Connected       | Disabled   |
| 38      | wg0                         | IPv4          | Connected       | Disabled   |

Y configuramos el IPv4 Forward en la interfaz con índice **38** mediante:

```
Set-NetIPInterface -ifIndex 38 -Forwarding Enabled
```



The screenshot shows a Windows PowerShell window titled "Seleccionar Administrador: Windows PowerShell". The command `Set-NetIPInterface -ifIndex 38 -Forwarding Enabled` is run, followed by `Get-NetIPInterface | select ifIndex, InterfaceAlias, AddressFamily, ConnectionState, Forwarding | Sort-Object -Property IfIndex | Format-Table`. The resulting table lists network interfaces with their properties:

| ifIndex | InterfaceAlias              | AddressFamily | ConnectionState | Forwarding |
|---------|-----------------------------|---------------|-----------------|------------|
| 1       | Loopback Pseudo-Interface 1 | IPv4          | Connected       | Disabled   |
| 1       | Loopback Pseudo-Interface 1 | IPv6          | Connected       | Disabled   |
| 7       | Conexión de red Bluetooth   | IPv6          | Disconnected    | Disabled   |
| 7       | Conexión de red Bluetooth   | IPv4          | Disconnected    | Disabled   |
| 8       | Ethernet                    | IPv4          | Disconnected    | Disabled   |
| 8       | Ethernet                    | IPv6          | Disconnected    | Disabled   |
| 16      | Wi-Fi                       | IPv4          | Connected       | Disabled   |
| 16      | Wi-Fi                       | IPv6          | Connected       | Disabled   |
| 18      | Ethernet 2                  | IPv6          | Connected       | Disabled   |
| 18      | Ethernet 2                  | IPv4          | Connected       | Disabled   |
| 38      | wg0                         | IPv6          | Connected       | Enabled    |
| 38      | wg0                         | IPv4          | Connected       | Enabled    |

- El último paso consiste en añadir una ruta a la máquina virtual (Develop) para indicarle la ruta a la dirección de la máquina host (Virtualizador, es decir, 192.168.56.1), de esta forma, podrá enrutar los paquetes recibidos de la VPN a través del Virtualizador hacia los servicios de la máquina virtual.

```
route -p add 192.168.10.0 mask 255.255.255.0 192.168.56.1
```

### Note

Otra forma de hacerlo, es establecer la puerta de enlace, de la máquina virtual, a la dirección del host.

```
PS C:\Windows\system32> route print
=====
Ilista de interfaces
 7...08 00 27 d0 cf 42 .....Intel(R) PRO/1000 MT Desktop Adapter
 4...08 00 27 ef 52 f5 .....Intel(R) PRO/1000 MT Desktop Adapter #2
 1.....Software Loopback Interface 1
=====

IPv4 Tabla de enrutamiento
=====
Rutas activas:
Destino de red      Máscara de red     Puerta de enlace   Interfaz   Métrica
 0.0.0.0            0.0.0.0           10.0.3.2          10.0.3.15    25
 10.0.3.0           255.255.255.0    En vínculo        10.0.3.15    281
 10.0.3.15          255.255.255.255  En vínculo        10.0.3.15    281
 10.0.3.255         255.255.255.255  En vínculo        10.0.3.15    281
 127.0.0.0           255.0.0.0          En vínculo        127.0.0.1    331
 127.0.0.1           255.255.255.255  En vínculo        127.0.0.1    331
 127.255.255.255   255.255.255.255  En vínculo        127.0.0.1    331
 192.168.10.0       255.255.255.0    192.168.56.1     192.168.56.101  26
 192.168.56.0       255.255.255.0    En vínculo        192.168.56.101  281
 192.168.56.101    255.255.255.255  En vínculo        192.168.56.101  281
 192.168.56.255    255.255.255.255  En vínculo        192.168.56.101  281
 224.0.0.0           240.0.0.0          En vínculo        127.0.0.1    331
 224.0.0.0           240.0.0.0          En vínculo        192.168.56.101  281
 224.0.0.0           240.0.0.0          En vínculo        10.0.3.15    281
 255.255.255.255   255.255.255.255  En vínculo        127.0.0.1    331
 255.255.255.255   255.255.255.255  En vínculo        192.168.56.101  281
 255.255.255.255   255.255.255.255  En vínculo        10.0.3.15    281
=====

Rutas persistentes:
 Dirección de red  Máscara de red  Dirección de puerta de enlace  Métrica
 192.168.10.0     255.255.255.0    192.168.56.1          1
=====

IPv6 Tabla de enrutamiento
=====
```

Con esto, podemos proceder a realizar las pruebas de conexión para confirmar que todo esta correcto.

## Pruebas de conexión

La primera prueba será comprobar que se tiene visibilidad de la máquina LAPTOP al Virtualizador (las dos IPs):

```
root@laptop:/etc/wireguard# ping -c 3 192.168.10.4
PING 192.168.10.4 (192.168.10.4) 56(84) bytes of data.
64 bytes from 192.168.10.4: icmp_seq=1 ttl=127 time=275 ms
64 bytes from 192.168.10.4: icmp_seq=2 ttl=127 time=307 ms
64 bytes from 192.168.10.4: icmp_seq=3 ttl=127 time=226 ms

--- 192.168.10.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 226.335/269.151/306.581/32.981 ms
root@laptop:/etc/wireguard#
root@laptop:/etc/wireguard# ping -c 3 192.168.56.1
PING 192.168.56.1 (192.168.56.1) 56(84) bytes of data.
64 bytes from 192.168.56.1: icmp_seq=1 ttl=126 time=346 ms
64 bytes from 192.168.56.1: icmp_seq=2 ttl=126 time=265 ms
64 bytes from 192.168.56.1: icmp_seq=3 ttl=126 time=289 ms

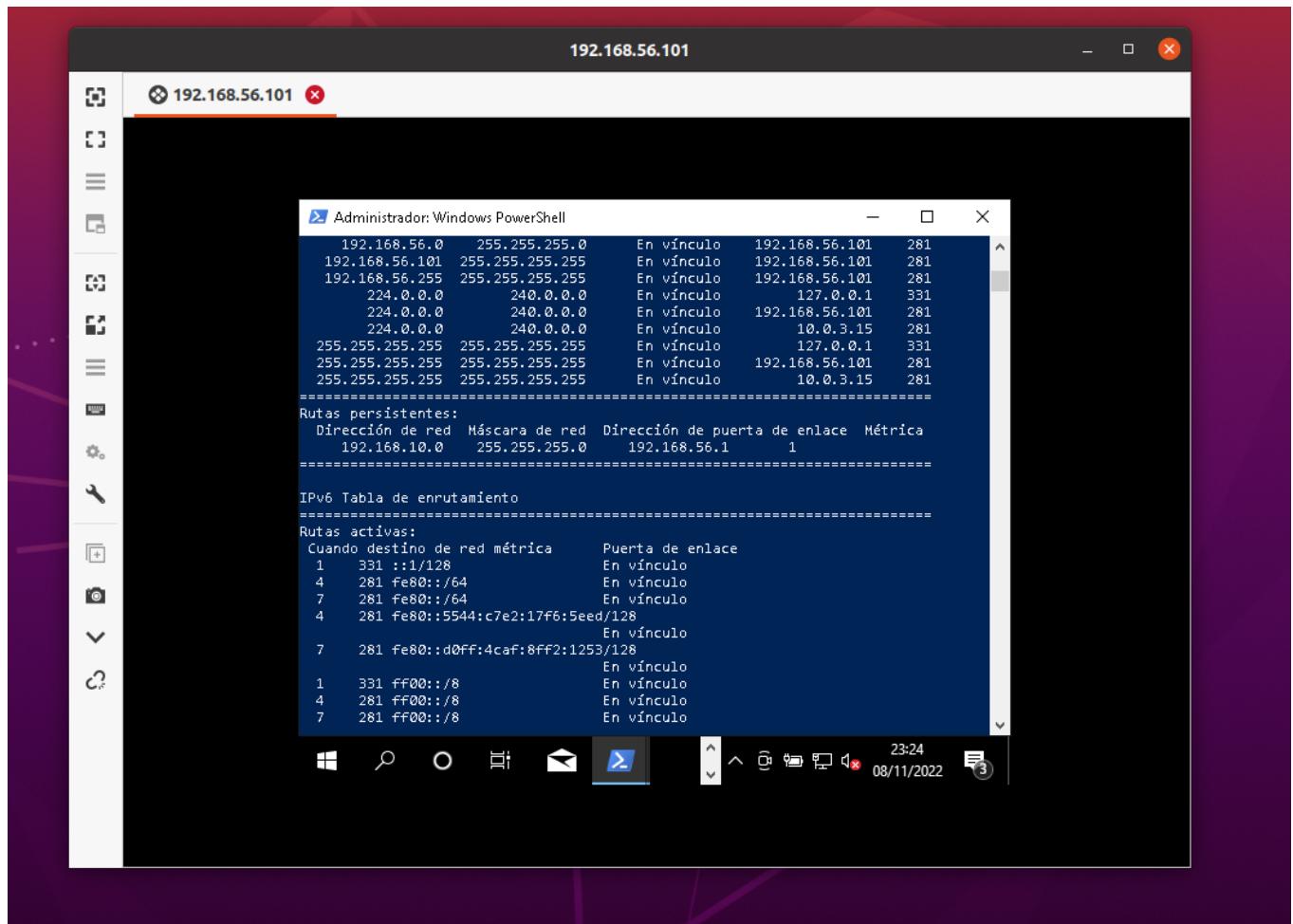
--- 192.168.56.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 264.850/299.760/345.856/34.003 ms
root@laptop:/etc/wireguard#
```

Tras verificar que todo esta correcto, se comprueba la visibilidad a la máquina Virtual **DEVELOP** desde LAPTOP:

```
root@laptop:/etc/wireguard# ping -c 3 192.168.56.101
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data.
64 bytes from 192.168.56.101: icmp_seq=1 ttl=126 time=299 ms
64 bytes from 192.168.56.101: icmp_seq=2 ttl=126 time=220 ms
64 bytes from 192.168.56.101: icmp_seq=3 ttl=126 time=345 ms

--- 192.168.56.101 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 220.251/288.089/345.282/51.595 ms
root@laptop:/etc/wireguard#
root@laptop:/etc/wireguard# nc -vz 192.168.56.101 3389
Connection to 192.168.56.101 3389 port [tcp/ms-wbt-server] succeeded!
root@laptop:/etc/wireguard#
```

Y con esto, nos conectamos vía RDP a la máquina Windows confirmando que el enrutamiento es correcto y se puede acceder a los servicios de las máquinas virtuales de un Cliente VPN desde otros clientes:



## Acceso Remoto a las máquinas de la organización

En esta sección se explicará, de forma breve y concisa cómo se podría realizar el acceso remoto a las máquinas de la Infraestructura.

### Acceso Remoto utilizando SSH

En primer lugar, para las máquinas Linux como puede ser el servidor VPS u otras máquinas Linux que conectemos a la Infraestructura, recomiendo utilizar **la autenticación basada en claves de SSH**

#### Note

Se recomienda utilizar la autenticación de clave pública en lugar de contraseña, debido a que, se evitan los ataques de fuerza bruta y se puede tener un mayor control de que personas tienen acceso a la máquina. Además, siendo VPS un

servidor en nube, y el servicio SSH estar expuesto a internet, se recomienda no tener autenticación basada en contraseñas, y si la hay, establecer una lo suficientemente robusta.

El primer paso consiste en generar el par de claves en la máquina que queremos utilizar para conectarnos, es decir, nuestro cliente. Debemos tener el cliente SSH instalado y, seguido de esto ejecutamos en una terminal:

```
ssh-keygen
```

Este comando nos generará un par de claves que se encuentran en la carpeta `~/.ssh/`. Ahí encontraremos dos archivos:

- `id_rsa` (clave privada)
- `id_rsa.pub` (clave publica)

Simplemente tenemos que copiar el contenido de la clave pública (`id_rsa.pub`) al archivo `.ssh/authorized_keys` del usuario que vamos a utilizar para conectarnos a la máquina remota.

En este caso, utilizaremos la cuenta `root` del VPS situado en el cloud. El contenido del archivo `/root/.ssh/authorized_keys` debería quedar de la siguiente forma:

```
ssh-rsa  
AAAAC3NzaC1yc2EAAAQABAAQgQCfP0hC5isvZN9lxLl+Aj2SKaGRFwJSp8lgxrV7rj//KIAD  
EDS9ApBQC1BkMI8zvIZFmPYV+tiWTtIWxR3H Aw25cHPTp4+BAM7HKk2Hl27xM0VR9Fw9nLl0z17V  
yD444o6bUW3VmXl4mXPuEJY4/190p1nnDOXnZ4JOMGRvYf14LZ0ryqlA2Hv6S1WW4APJIMB6sJAE  
v5JX7UBHvnirnXjeS04WjxwNvP7peVm iqCcqcVgGgz0+v8SNQclD+TpdyHdXyCpwVE2rFNzrv68o  
oYumCxZ427jWJHeEfqqYy0efvCmI6hYi0iM+SR9zBlvngIz00Qx7jGzpWJJXEaofrkHTTo9Vptgg  
LA9kk8NicMBKxLHKMyFh9I2FfYXoL4Q07MrCbbtag78+xPEP75NNrtWVusPHBboGiva5ul6K4B6R  
ZJon/F1kJF5t9khzBVRcUHtGsPNo2pGPv2cCnYodmPKA1qdslhbx5Q+wEWm3IntS/VJ+JVWG GdsM  
mX+M7sVJHHs=
```

Una vez hecho esto, simplemente procedemos a autenticarnos desde la máquina cliente con el comando:

```
ssh root@<ip-servidor-vps>
```

Y comprobaremos que nos autenticamos sin introducir la contraseña del usuario en cuestión. Esto sucede debido a que se está utilizando el modelo de autenticación por claves para validar la identidad del usuario.

## Note

En caso de que no queramos recordar la IP del servidor VPS, para facilitarlo un poco, se puede crear el siguiente archivo de configuración SSH en la máquina cliente:

`~/.ssh/config` con el siguiente contenido (o añadirlo en caso de ya existir):

```
Host vps-cloud-server
  Hostname ip-servidor-vps
  User root
  Port 22
```

Para después autenticarse mediante el comando:

```
ssh vps-cloud-server
```

```
(base) secu@x11:~$ tail -4 .ssh/config
Host vps-cloud-server
  Hostname 165.22.71.211
  User root
  Port 22
(base) secu@x11:~$ ssh vps-cloud-server
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-52-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

System information as of Wed Nov  9 17:30:34 UTC 2022

System load:      0.080078125      IPv4 address for br_hackmd: 172.18.0.1
Usage of /:       7.7% of 48.27GB    IPv4 address for docker0:   172.17.0.1
Memory usage:    25%
Swap usage:      0%
Processes:       119                 IPv4 address for eth0:    165.22.71.211
Users logged in: 1                  IPv4 address for eth0:    10.19.0.5
                                         IPv4 address for eth1:    10.114.0.2
                                         IPv4 address for wg0:    192.168.10.1

11 updates can be applied immediately.
1 of these updates is a standard security update.
To see these additional updates run: apt list --upgradable

Last login: Wed Nov  9 11:52:41 2022 from 66.81.160.54
root@vps-cloud-server:~#
```

## Acceso Remoto utilizando RDP

En segundo lugar, se explicará (brevemente) el acceso a las máquinas Windows a través de la VPN. Al igual que para las máquinas Linux utilizábamos SSH para acceder, en el caso de Windows utilizaremos RDP.

Se ha preferido RDP frente a otras soluciones debido a que viene incorporado en Windows y simplemente debe habilitarse para poder utilizarlo. RDP nos permitirá tanto acceder gráficamente a la máquina Windows, como también realizar acciones como el intercambio de archivos entre cliente y servidor.

Para acceder desde un cliente Windows a otro utilizando RDP, se puede utilizar directamente el cliente RDP de Windows: mstsc.exe. Por otro lado, si se desea conectarse desde un cliente Linux, se pueden utilizar diferentes herramientas, para este caso se va a explicar el uso de Remmina.



Se ha elegido Remmina frente a otras soluciones pues, por defecto, no solo soporta RDP sino que también SSH o VNC. Tiene una interfaz gráfica muy sencilla y efectiva. Otra funcionalidad interesante de Remmina es que te permite agrupar los perfiles de conexión por "Grupos", lo cual permite tener una mayor organización y control.

En Remmina es posible crear y guardar perfiles de conexión, los cuales facilitan el acceso a máquinas pues, en el próximo uso, ya se dispone de la información relacionada con la conexión y simplemente te tienen que introducir las credenciales.

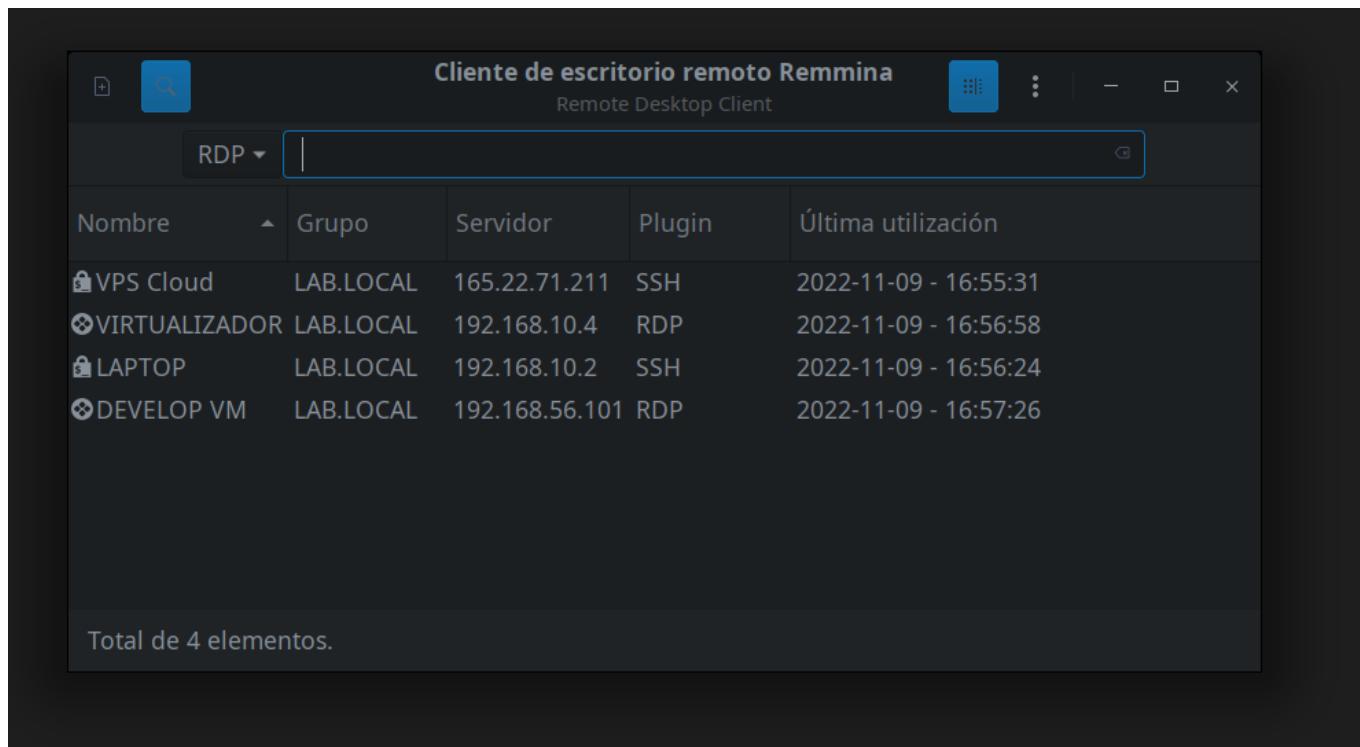
Estos perfiles de conexión son almacenados en una carpeta de Remmina:

```
~/.local/share/remmina/
```

### Note

Esta funcionalidad sugiere que, si disponemos de múltiples máquinas en la Infraestructura (tanto Windows como Linux), podemos generalizar el acceso remoto añadiendo este directorio a nuestro servicio de control de versiones (Gitea). De esta forma, los integrantes de la organización tendrán los perfiles de acceso a cada una de las máquinas de la Infraestructura y podrán realizar los cambios pertinentes y verlos reflejados.

Tras la creación de distintos perfiles de conexión, podemos obtener una vista como la siguiente:



The screenshot shows the Remmina Remote Desktop Client interface. The title bar reads "Cliente de escritorio remoto Remmina" and "Remote Desktop Client". The main window displays a table of saved connections with the following data:

| Nombre        | Grupo     | Servidor       | Plugin | Última utilización    |
|---------------|-----------|----------------|--------|-----------------------|
| VPS Cloud     | LAB.LOCAL | 165.22.71.211  | SSH    | 2022-11-09 - 16:55:31 |
| VIRTUALIZADOR | LAB.LOCAL | 192.168.10.4   | RDP    | 2022-11-09 - 16:56:58 |
| LAPTOP        | LAB.LOCAL | 192.168.10.2   | SSH    | 2022-11-09 - 16:56:24 |
| DEVELOP VM    | LAB.LOCAL | 192.168.56.101 | RDP    | 2022-11-09 - 16:57:26 |

Total de 4 elementos.

## Estrategia de Backups

Finalmente, el último paso consiste en diseñar un buen sistema de Backups que nos permita asegurar la información contenida en los diferentes servicios de la Infraestructura.

En ocasiones, los despliegues con Docker generan una cierta sensación de inseguridad y aportar preocupación ante la posibilidad de romperse el contenedor y perderse la información.

A continuación se explica el diseño y herramientas que nos permitirá realizar copias de seguridad periódicas y guardar la información a salvo.

## Arquitectura y Automatización de Backups

En la sección de "Servicio de Orquestación de Aplicaciones - Docker", se definió una jerarquía de directorios para las aplicaciones contenerizadas. Esta estructura fue diseñada con el motivo de mantener los contenedores y su información de forma organizada, pero sobretodo, para poder realizar Backups de los contenedores de forma automatizada.

El proceso de Backups de los contenedores Docker se divide en 3 bloques:

1. Se itera sobre el directorio de los contenedores `/srv/containers/` y se exporta las imágenes de cada aplicación.
2. Después, se itera sobre el directorio de datos `/srv/data/` (de las aplicaciones) y se realiza una copia de los volumenes de cada aplicación.
3. Finalmente, se comprime y guarda cada backup de contenedor y volumen por cada aplicación (pasos 1 y 2).

Al acabar el proceso, se descargan los ficheros resultantes a un equipo local, y se almacenan rotativamente (esto se explicará a continuación).

Puntos de interes:

- Se exportan las imágenes de los contenedores pues, en caso de una migración, simplemente se arrastra el backup de la imagen y se carga con Docker. Por otro lado, los datos de los volúmenes, solo se han de volver a copiar a la ruta donde estaban y listo.
- Cada aplicación contenerizada debe tener un archivo `docker-compose.yml` en el directorio raíz de su contenedor: `/srv/containers/<app>/docker-compose.yml`, de esta forma, el script que realiza los backups, podrá procesar el archivo de orquestación y exportar las imágenes de cada contenedor que levante la aplicación.

El script utilizado para realizar los Backups es el siguiente:

```
#!/bin/bash

containers_path="/srv/containers"
volumes_path="/srv/data"
backups_path="/srv/backups"

# Iterate over containers path and export images of every service
```

```

for service in $(ls $containers_path); do

    docker_compose_path="$containers_path/$service/docker-compose.yml"
    if test -f $docker_compose_path; then

        mkdir -p $backups_path/$service
        outfile="${backups_path}/${service}/${service}_images.tar"

        echo -e "[+] Exporting images from $service..."

        # Save images from docker-compose file
        docker save -o $outfile $(docker compose -f $docker_compose_path
config | awk '{if ($1 == "image:") print $2;}') &>/dev/null
    fi
done

# Iterate over volumes path and store contents of every service
for service in $(ls $volumes_path); do

    echo -e "[+] Store volumes from $service..."
    mkdir -p $backups_path/$service
    tar -czvf "${backups_path}/${service}/${service}_volumes.tar.gz"
    $volumes_path/$service &>/dev/null
done

# Iterate over backups path and pack every service files in path
for service in $(ls $backups_path); do

    echo -e "[+] Packing backups files from $service..."
    date=$(date '+%Y%m%d_%H%M%S')
    tar -czvf "${backups_path}/${date}_${service}_backup.tar.gz"
    $backups_path/$service &>/dev/null

    echo -e "[+] Done! Clearing backups files from $service..."
    rm -rf $backups_path/$service
done

```

Lo situamos en la ruta: `/srv/` y le damos permisos con: `chmod 700 backup_containers.sh`

```
root@vps:/srv# ls -la
total 24
drwxr-xr-x  5 root root 4096 nov  9 18:10 .
drwxr-xr-x 20 root root 4096 jul 11 18:03 ..
-rwx----- 1 root root 1380 nov  9 18:10 backup_containers.sh
drwxr-xr-x  2 root root 4096 oct 16 09:48 backups
drwxr-xr-x  9 root root 4096 nov  7 17:47 containers
drwxr-xr-x  9 root root 4096 nov  1 12:47 data
root@vps:/srv#
```

### Note

Si no existe el directorio `/srv/backups` debemos crearlo para que el script pueda situar ahí los archivos de backups.

Al ejecutarlo, el script comienza el proceso:

```
root@vps:/srv# ./backup_containers.sh
[+] Exporting images from gitea...
[+] Exporting images from hackmd...
[+] Exporting images from jitsi...
[+] Exporting images from syspass...
[+] Exporting images from wekan...
```

Tras exportar las imágenes, procederá a comprimir los volúmenes y dejará todos los archivos en `/srv/backups` quedando como el siguiente ejemplo:

```
root@vps:/srv# ls -lh /srv/backups/
total 377M
-rw-r--r-- 1 root root 102M nov  9 18:14 20221109_181400_gitea_backup.tar.gz
-rw-r--r-- 1 root root 5,1M nov  9 18:14 20221109_181409_hackmd_backup.tar.gz
-rw-r--r-- 1 root root 259M nov  9 18:14 20221109_181409_jitsi_backup.tar.gz
-rw-r--r-- 1 root root 1,8M nov  9 18:14 20221109_181432_syspass_backup.tar.gz
-rw-r--r-- 1 root root 4,4M nov  9 18:14 20221109_181432_wekan_backup.tar.gz
-rw-r--r-- 1 root root 5,7M nov  9 18:14 20221109_181432_wikijs_backup.tar.gz
-rw-r--r-- 1 root root  321 nov  9 18:14 20221109_181432_zulip_backup.tar.gz
root@vps:/srv#
```

En este punto, nos queda automatizar este proceso y traernos las copias a local. Para esto, se definirá un script que ejecutaremos localmente que realizará los siguientes pasos:

1. Ejecutará el script de backups a través del SSH en VPS
2. Se copiará los backups resultantes a local a través del SSH
3. Eliminará los backups locales que llevan más de 5 días generados
4. Limpiará los archivos de backups generados en VPS

Este script tendría, en mi caso, la siguiente apariencia:

```
#!/bin/bash

ssh vps "/srv/backup_containers.sh"
scp vps:/srv/backups/* ~/backups_infra/
find ~/backups_infra/ -type f -mtime +5 -delete
ssh vps "rm -f /srv/backups/*"
```

### Note

Obviamente esto se debe modificar a gusto propio y decidir que se hace con los backups, cómo se almacenan y hasta cuando.

Al ejecutarse el script podemos seguir todo el proceso de backups (se recomienda volcar el output a un log para poder analizarlo en caso de error):

```
(base) secu@x11:~$ ./auto_backup.sh | tee -a auto_backup.log
[+] Exporting images from gitea...
[+] Exporting images from hackmd...
[+] Exporting images from jitsi...
[+] Exporting images from syspass...
[+] Exporting images from wekan...
[+] Exporting images from wikijs...
[+] Exporting images from zulip...
[+] Store volumes from gitea...
[+] Store volumes from hackmd...
[+] Store volumes from jitsi...
[+] Store volumes from syspass...
[+] Store volumes from wekan...
[+] Store volumes from wikijs...
[+] Store volumes from zulip...
[+] Packing backups files from gitea...
[+] Done! Clearing backups files from gitea...
[+] Packing backups files from hackmd...
[+] Done! Clearing backups files from hackmd...
[+] Packing backups files from jitsi...
```

Finalmente, tendríamos el directorio de backups local con el siguiente contenido:

```
(base) secu@x11:~$ ls -lh backups_infra/
total 377M
-rw-r--r-- 1 secu secu 102M nov  9 19:34 20221109_183410_gitea_backup.tar.gz
-rw-r--r-- 1 secu secu 5,1M nov  9 19:34 20221109_183421_hackmd_backup.tar.gz
-rw-r--r-- 1 secu secu 259M nov  9 19:34 20221109_183422_jitsi_backup.tar.gz
-rw-r--r-- 1 secu secu 1,8M nov  9 19:34 20221109_183447_syspass_backup.tar.gz
-rw-r--r-- 1 secu secu 4,5M nov  9 19:34 20221109_183447_wekan_backup.tar.gz
-rw-r--r-- 1 secu secu 5,7M nov  9 19:34 20221109_183447_wikijs_backup.tar.gz
-rw-r--r-- 1 secu secu 322 nov  9 19:34 20221109_183447_zulip_backup.tar.gz
(base) secu@x11:~$
```

El último paso a realizar sería crear un cronjob que ejecutase este script cada día, a las 04:00 de la mañana (por ejemplo). El cronjob sería el siguiente:

```
0 4 * * * /home/secu/auto_backup.sh | tee -a /home/secu/auto_backup.log
```

Y con esto, se finaliza la guía. Espero que haya sido de utilidad, que pueda haber aportado ideas acerca de la Infraestructura y organización de recursos. Es posible que haya algún fallo o no se haya profundizado lo suficiente en alguna sección, para ese caso, no dudes en escribirme por Twitter (@secu\_x11) o Telegram (@Secury).

## Referencias

- <https://www.digitalocean.com/community/tutorials/how-to-set-up-wireguard-on-ubuntu-20-04>
- <https://computingforgeeks.com/install-and-configure-dnsmasq-on-ubuntu/>
- <https://compilar.kryptonsolid.com/como-crear-un-certificado-ssl-autofirmado-para-nginx-en-ubuntu-18-04/>
- <https://www.ochobitshacenunbyte.com/2021/04/26/netplan-configurar-la-red-en-ubuntu-20-04/>