Laboratorio de Redes y Sistemas Operativos
Trabajo Práctico Final

Temática: Instalación y puesta en funcionamiento de un servicio de red.

Integrantes:

- Banegas Nicolás
- Benitez Uriel
- Rodriguez Franco

Profesor: Di Biase José Luis

Descripción del proyecto

Para el presente trabajo hemos elegido instalar y probar ***Synapse***, que es la implementación de un servidor Matrix de mensajería programador en Python/Twisted y Rust.

Synapse es mantenido actualmente por la empresa Element, que produce soluciones de software open - source basadas en Matrix. La principal diferencia entre utilizar Synapse con otro servicio de mensajería masivo, como Whatsapp, es que Synapse es descentralizado, dándole la posibilidad a cada usuario en particular de tener su propio servidor proveedor de mensajes. Esto genera “soberanía digital” para los usuarios.

Hardware requerido y hardware utilizado

Esto no es una medida fija, depende de la cantidad de usuarios que se hosteen en el servidor Synapse.
En nuestro caso, las especificaciones de los servidores utilizados fueron:

- whatsapp2 (servidor Synapse 1) :
- S.O - Ubuntu 24.04.3 LTS (Noble Numbat), virtualizado (VirtualBox)
- Memoria RAM 8GB
- Disco duro 50GB
- Procesador I5-8265U 4 núcleos
- Gráfica integrada

- whatsapp 3 (servidor Synapse 2):
- S.O - Ubuntu 24.04.3 LTS (Noble Numbat), virtualizado (VirtualBox)
- Memoria RAM 8GB
- Disco duro 50GB
- Procesador: AMD Ryzen 7 5700U with Radeon Graphics
- Cantidad núcleos: 4

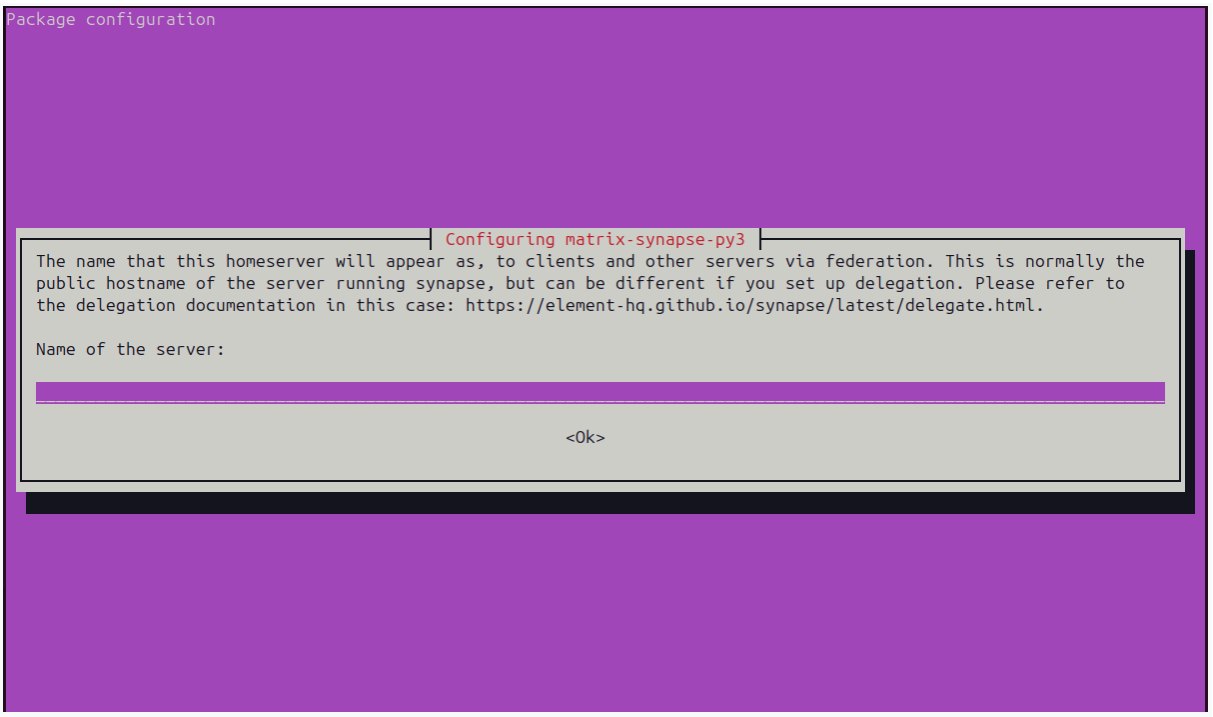
Guía de instalación

Hay diferentes maneras de instalar el programa servidor de *Synapse* en nuestro dispositivo. En este caso nosotros optamos por la opción que ofrece matrix.org con sus paquetes para Debian/Ubuntu.

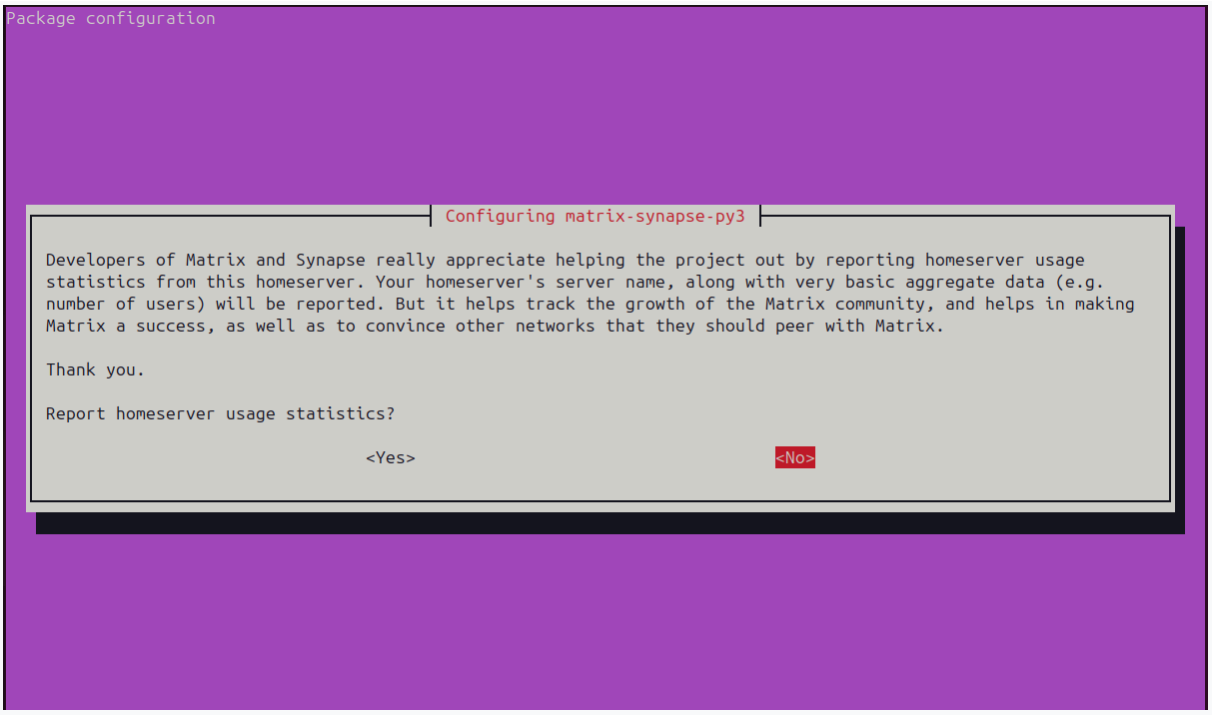
Entonces, comencemos…
Abrimos la terminal y ejecutamos los siguientes comandos:

```
sudo apt install -y lsb-release wget apt-transport-https
sudo wget -O /usr/share/keyrings/matrix-org-archive-keyring.gpg
https://packages.matrix.org/debian/matrix-org-archive-keyring.gpg
echo "deb [signed-by=/usr/share/keyrings/matrix-org-archive-keyring.gpg]
https://packages.matrix.org/debian/ $(lsb_release -cs) main" |
    sudo tee /etc/apt/sources.list.d/matrix-org.list
sudo apt update
sudo apt install matrix-synapse-py3
```

Esto instalará Synapse y abrirá una interfaz donde debemos ingresar el nombre de nuestro servidor. Tener en cuenta que este nombre no se podrá cambiar más adelante.



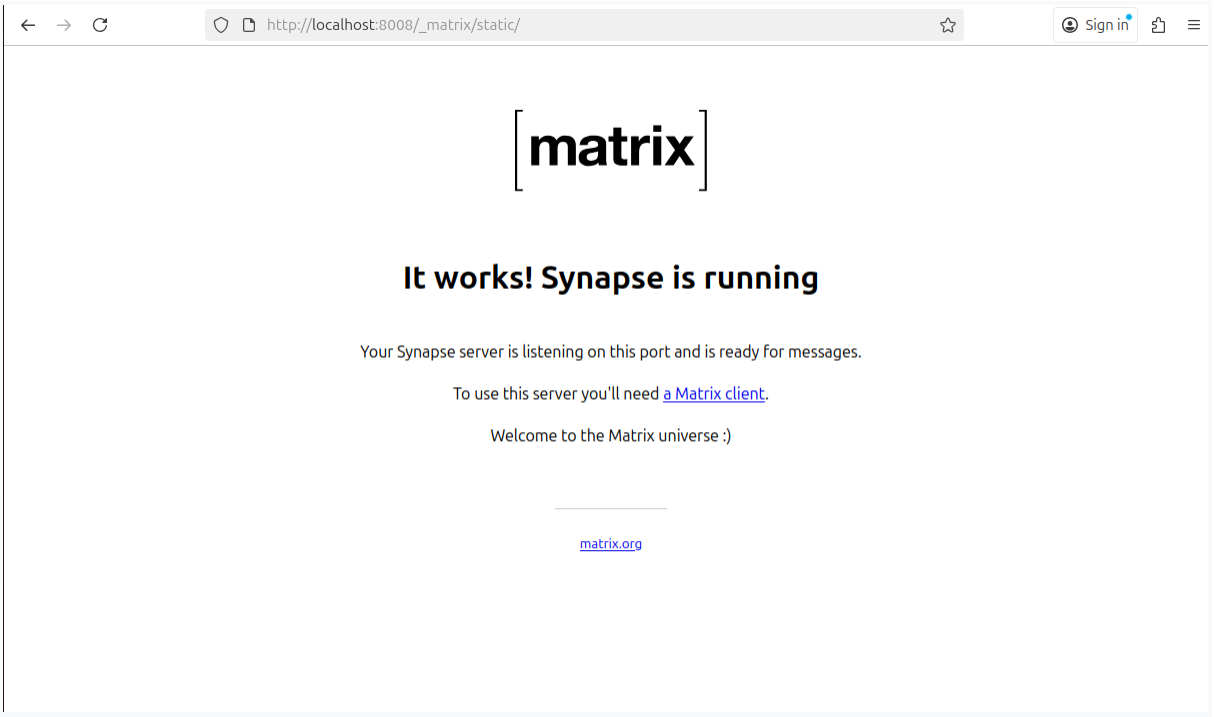
Luego de indicar el nombre del servidor, se abrirá otra pestaña donde nos invitan a compartir un feedback con Matrix, ponemos que NO y seguimos…



Ahora para verificar que la instalación haya salido bien debemos usar el programa service, tal que:

```
service matrix-synapse status
```

Si hicimos todo bien, debe aparecer el servicio activo (running).
Otra opción de verificar la correcta instalación es en nuestro navegador web, haciendo una petición a la URL localhost:8008, ya que el servidor Synapse si está activo empieza escuchando por defecto en el puerto 8008.



Inicialmente, el archivo de configuración del servidor Synapse utiliza una base de datos SQLite. Es recomendable utilizar PostgreSQL,ya que es más eficiente para el caso de mensajería masiva.

Entonces debemos instalar PostgreSQL en nuestro S.O, para esto, ejecutamos lo siguiente en la terminal:

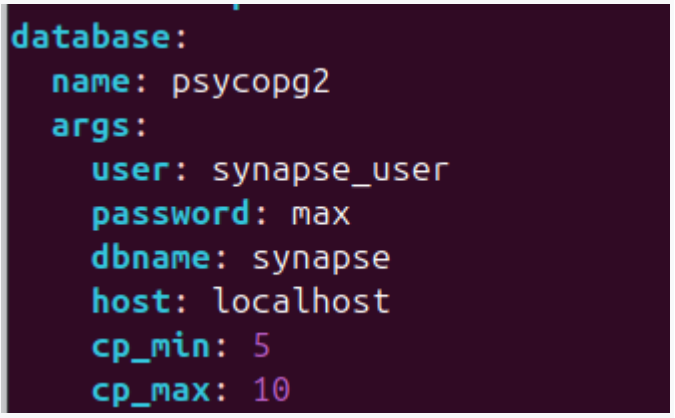
```
sudo apt install postgresql
```

Ahora debemos cambiar el archivo de configuración del servidor. Lo hacemos de la siguiente manera:

Vamos a la ruta por defecto del archivo de configuración, la cual debe ser /etc/matrix-synapse/homeserver.yaml si lo instalamos a través de los paquetes de Debian/Ubuntu, y lo abrimos con nuestro editor de texto.

```
sudo vim homeserver.yaml
ó
sudo nano homeserver.yaml
```

Ahora, borramos la configuración ***database*** en el archivo homeserver.yaml (el de configuración del servidor Synapse) ya que es la configuración estándar que utiliza SQLite. Y ponemos la siguiente configuración



El atributo name especifica el motor de bases de datos a utilizar. En nuestro caso, como estamos utilizando PostgreSQL, debemos utilizar psycopg2.
El atributo args sirve para listar los argumentos para la conexión a la base de datos a través de psycopg2.
El atributo user especifica el usuario de la base de datos.
El atributo password es la contraseña del usuario de la base de datos.
El atributo dbname indica el nombre de la base de datos que utilizará nuestro servidor.
El atributo host indica la IP del dispositivo donde está alojada la base de datos.
cp_min y cp_max establecen la cantidad mínima de conexiones que Synapse mantiene abierta siempre, y la cantidad máxima de conexiones simultáneas que se pueden abrir, respectivamente.
En nuestro caso, son valores pequeños porque es un servidor de pruebas, en un servidor público masivo deben ser valores mayores.

Después de haber modificado el archivo de configuración del servidor, Synapse necesita modificar la signing.key del servidor. Esta clave privada es usada por el servidor para firmar mensajes y funciona como identificador único del servidor.
Entonces el servicio, luego de modificar el archivo de configuración, debe ser reiniciado para validar los cambios:

```
service matrix-synapse restart
```

Ahora, en este punto lo que sucede es que NO se puede reiniciar el servicio ya que cuando Synapse intenta modificar la clave tras haber realizado los cambios en homeserver.yaml, no tiene los permisos para hacerlo. Entonces tenemos que cambiar el dueño de la carpeta correspondiente al servicio para que pueda hacerlo. Lo hacemos de la siguiente manera:

```
sudo chown -R matrix-synapse:matrix-synapse /etc/matrix-synapse
```

Esto hace que la carpeta y todos los archivos dentro sean del dueño y grupo matrix-synapse.

Finalmente, intentamos reiniciar el servicio nuevamente y verificamos que efectivamente esté corriendo.

```
service matrix-synapse status
```

```
root@uriel-benitez-VirtualBox:/etc/matrix-synapse# service matrix-synapse restart
root@uriel-benitez-VirtualBox:/etc/matrix-synapse# service matrix-synapse status
● matrix-synapse.service - Synapse Matrix homeserver
     Loaded: loaded (/usr/lib/systemd/system/matrix-synapse.service; enabled; preset: enabled)
     Active: active (running) since Mon 2025-12-08 18:34:46 -03; 11s ago
    Process: 29636 ExecStartPre=/opt/venvs/matrix-synapse/bin/python -m synapse.app.homeserver >
   Main PID: 29644 (python)
      Tasks: 16 (limit: 9180)
     Memory: 89.8M (peak: 90.7M)
        CPU: 3.361s
```

Teniendo todo esto, nuestro servidor Synapse ya está en funcionamiento. Ahora pasamos a la instalación del cliente Element.

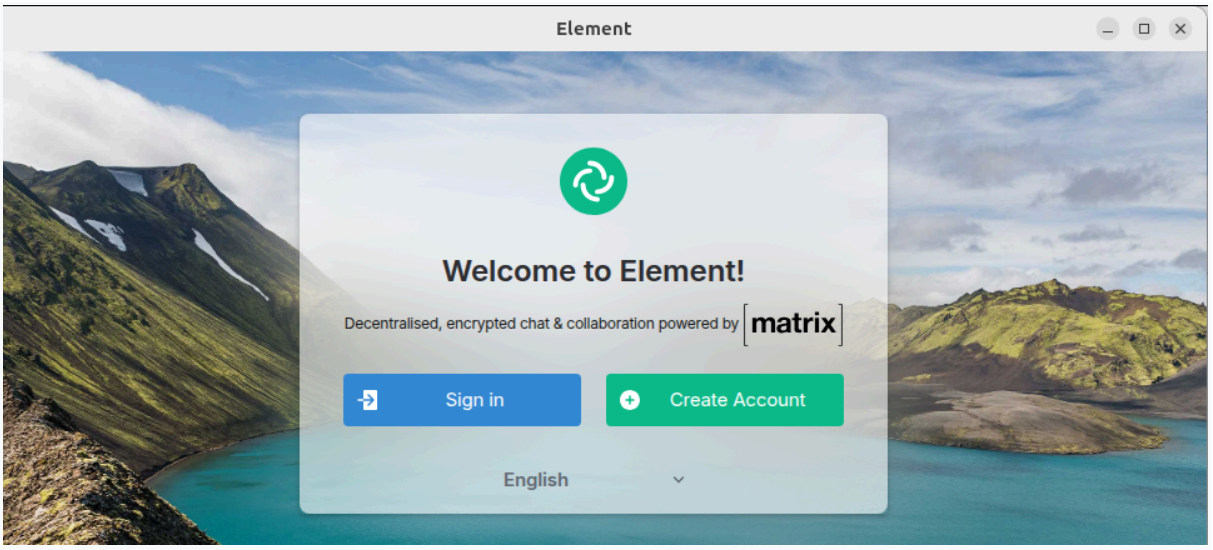
Para instalarlo en Ubuntu, corremos en la terminal los siguientes comandos:

```
sudo apt install -y wget apt-transport-https

sudo wget -O /usr/share/keyrings/element-io-archive-keyring.gpg
https://packages.element.io/debian/element-io-archive-keyring.gpg

echo "deb [signed-by=/usr/share/keyrings/element-io-archive-keyring.gpg]
https://packages.element.io/debian/ default main" | sudo tee
/etc/apt/sources.list.d/element-io.list

sudo apt update

sudo apt install element-desktop
```

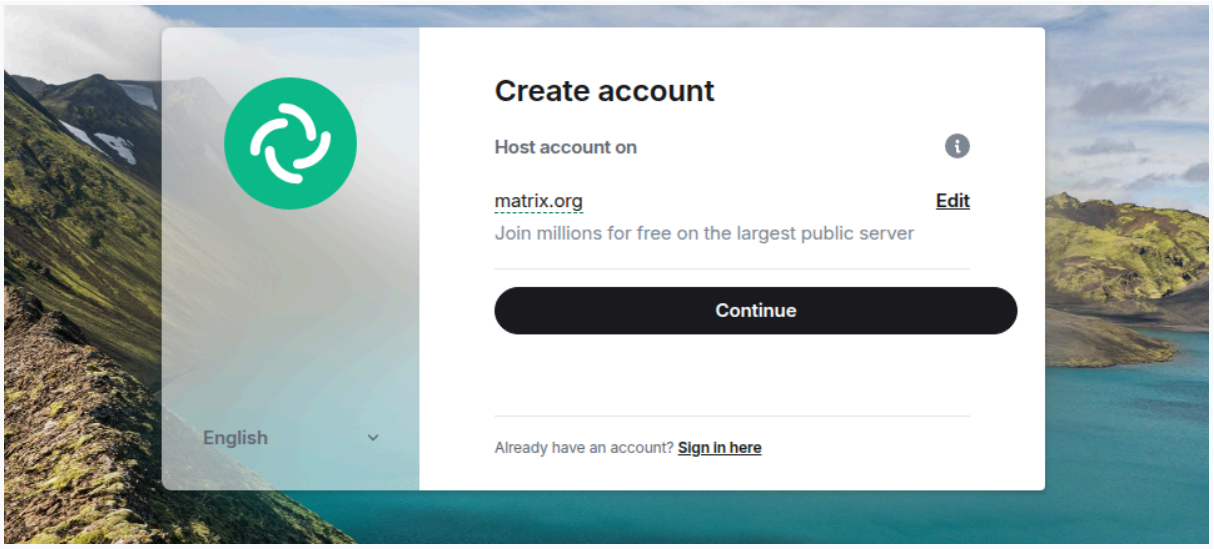
Una vez instalado, lo abrimos con:

```
element-desktop
```

Seleccionamos la opción de crear cuenta…

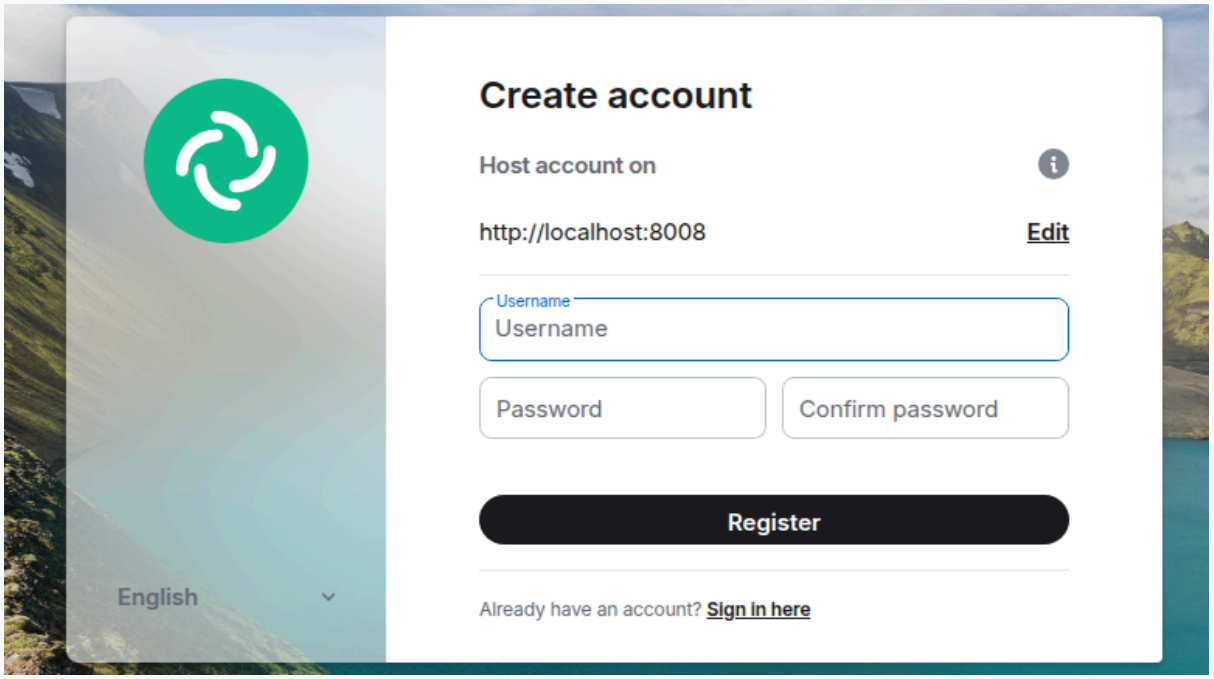


Y ahora editamos el servidor que hosteara nuestra cuenta, ya que queremos que la cuenta sea hosteada en nuestro servidor local…



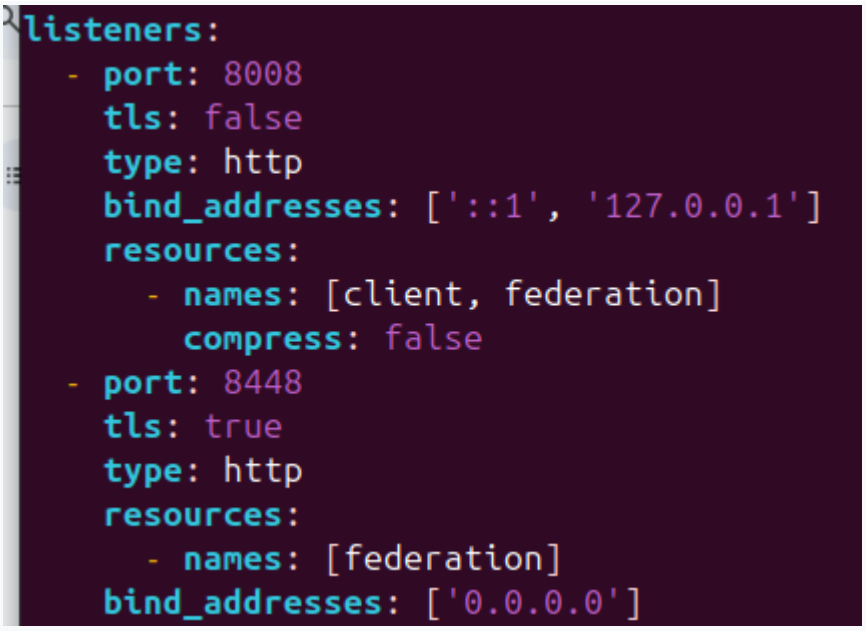
Ahora la interfaz nos indicará que nuestro servidor no está habilitado para registrarse. Entonces debemos ir a homeserver.yaml y agregar las siguientes líneas:

```
enable_registration:true
registration_requires_token:false
enable_registration_without_verification:true
```



Creamos nuestra cuenta con las respectivas credenciales e iniciamos sesión…

Ahora, para poder comunicarnos con otro servidor Synapse, debemos habilitar el puerto de federación. Para esto tenemos que volver a homeserver.yaml y agregar como listener el puerto 8448 con TLS activado para el cifrado de mensajes.



```
listeners:
  - port: 8008
    tls: false
    type: http
    bind_addresses: ['::1', '127.0.0.1']
    resources:
      - names: [client, federation]
        compress: false
  - port: 8448
    tls: true
    type: http
    resources:
      - names: [federation]
    bind_addresses: ['0.0.0.0']
```

También debemos crear una whitelist de los servidores con los cuales queremos federar, y una blacklist de quienes están bloqueados, en nuestro caso nadie…

```
federation_domain_whitelist:
  - "whatsapp2"
  - "whatsapp3"
federation_ip_range_blacklist: []
```

Luego, creamos los certificados para el servidor con el programa openssl

```
openssl req -new -newkey rsa:4096 -sha256 -days 365 \
  -nodes -x509 \
  -keyout server.key \
  -out server.crt
```

Luego, debemos especificar la ruta donde debe se deben leer los certificados y claves generadas en el archivo de configuración, tal que:

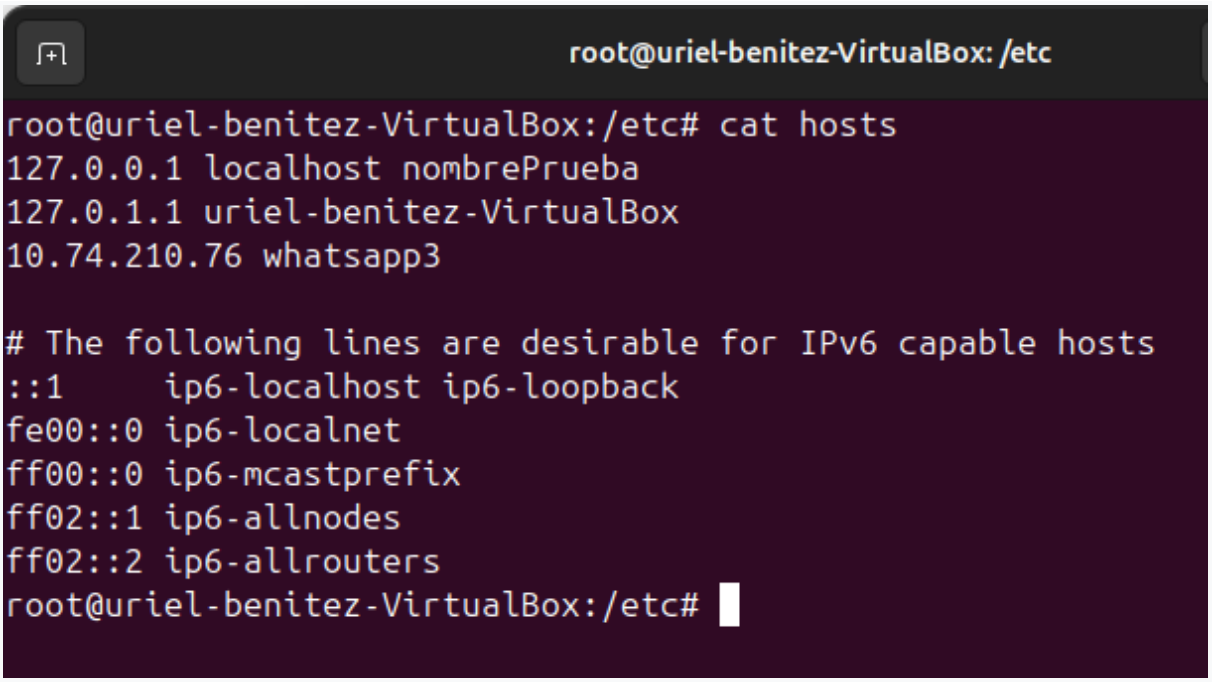
```
tls_certificate_path: "/etc/matrix-synapse/server.crt"
tls_private_key_path: "/etc/matrix-synapse/server.key"
```

Esto lo que genera son los certificados TLS self-signed y la clave privada del servidor con la cual firma eventos. Ahora, Element necesita que estos certificados sean firmados por una CA (Certificate Authority), es decir un tercero que verifica que el servidor Synapse efectivamente es quien dice ser. El problema es que para generar estos “certificados reales” necesitamos un dominio público, lo cual no tenemos. Entonces lo que tenemos que hacer para poder federar en nuestra LAN es desactivar la verificación de certificados y claves de los servidores con los que vamos a federar.

Para hacer esto debemos cambiar lo siguiente en homeserver.yaml:

```
federation_verify_certificates: false // Para que evite la validación de certificados, SOLO válido en un ambiente de prueba donde sabemos que podemos confiar en el otro servidor.

trusted_key_servers:
  - server_name: "whatsapp3" // El fqdn del servidor
    accept_keys_insecurely: true // Para que acepte las claves de forma insegura
```

Como último paso, debemos agregar en el archivo etc/hosts la línea para que Synapse pueda resolver el dominio del servidor al cual queremos comunicarnos cuando creamos una sala en el cliente Element.



En este caso, el servidor Synapse al cual queremos mandarle un mensaje es whatsapp3. Cabe aclarar que este archivo debe ser modificado cada vez que cambiemos de red.

De esta manera, queda totalmente funcional el servicio de Synapse para mensajería. Simplemente ahora en nuestro cliente debemos crear una sala con el usuario con el cual queremos chatear. El formato para contactarnos con otro usuario es *@nombre_usuario:fqdn_servidor*.

Problemáticas encontradas y soluciones aplicadas

La principal problemática que tuvimos fue al momento de federar entre 2 servidores. Inicialmente probamos registrar y comunicarnos con usuarios que están hosteados en el mismo servidor Synapse, pero para agregar una mayor funcionalidad implementamos otro servidor con la misma configuración en otro dispositivo en la misma red LAN. A raíz de esto, surgió el problema de los certificados TLS, punto en el cual tuvimos que recurrir a ayuda de Inteligencia Artificial (ChatGPT) para poder depurar los problemas que teníamos, básicamente que no podíamos establecer la conexión con el otro servidor, nunca le llegaban los mensajes, y resolverlos concretamente. Gracias a

la IA entendimos por qué no alcanzaba con simplemente generar los certificados self-signed (descrito anteriormente), y por esto desactivamos las verificaciones pertinentes en ambos servidores (también descrito en la sección anterior).

Fuentes y recursos utilizados

- Manual de Synapse (documentación para entender cómo configurar el homeserver)
- PostgreSQL (para su instalación)
- Repositorio de Element
- Element Client (para efectivamente utilizar el servidor desplegado)
- ChatGPT (para depurar errores y consultas técnicas sobre temas como federación entre servidores y certificados TLS)