

Mandant : École polytechnique fédérale de Lausanne

Chef de projet : M. Luc Venries

Responsable : M. Alain Bron

Mandataire : M. Nathan Ernst

Du 21 août 2023 au 26 septembre 2023

[Introduction 1](#_Toc710773466)

[Le projet 2](#_Toc757372787)

[Le protocole Matrix 2](#_Toc1913860537)

[Matrix qu’est-ce que c’est ? 2](#_Toc64730479)

[Matrix – Le fondamental 2](#_Toc1223706525)

[Les rooms 3](#_Toc1716951664)

[Les bridges 4](#_Toc1407267707)

[Les spaces 5](#_Toc160761918)

[Les APIs 7](#_Toc1876969957)

[Principe de communication 8](#_Toc364734234)

[Matrix – Architecture 9](#_Toc2065322888)

[Les Acteurs 10](#_Toc1008070005)

[Serveur 11](#_Toc256261525)

[Environnement 12](#_Toc730330714)

[Synapse 12](#_Toc1269754827)

[Serveur d’identification 12](#_Toc1848084118)

[Installation de Synapse 12](#_Toc358684937)

[PostgreSQL 16](#_Toc1991276029)

[Configuration de la base de données 17](#_Toc2018617711)

[Docker 18](#_Toc1692738408)

[Installation 18](#_Toc2107056433)

[Installation sans automatisation 22](#_Toc957213065)

[Synapse 23](#_Toc1930108654)

[PostGres 24](#_Toc1088602708)

[Reverse Proxy 26](#_Toc1099155744)

[Glossaire 33](#_Toc490459972)

# Introduction

Les collaborateurs de l’École polytechnique fédérale de Lausanne − ci-après EPFL − utilisent différents outils pour communiquer. Les produits Cisco (Cisco Jabber) sont utilisés pour la téléphonie (softphones) et fournissent également une messagerie (protocole XMPP). Bien que certaines salles de visioconférences utilisent Cisco WebEx, l’outil de vidéoconférence officiel est Zoom, qui lui aussi propose une messagerie instantanée. Dans les différents services et unités de l’EPFL, on trouve aussi Slack, Mattermost, Telegram, Signal, Discord ou WhatsApp et bien que certains d’entre-eux proposent aussi la téléphonie ou la vidéo-conférence, ils sont principalement utilisés pour les fonctionnalités de messageries instantanée (chats).

Le mandant, l’EPFL, aimerai agréger / fédérer ses différents flux de communication tels que produits par les applications Cisco *Jabber* ou *WebEx*, ou encore *Zoom*, via le protocole ouvert Matrix

# Le projet

Ce projet consiste à étudier et mettre en œuvre le protocole Matrix et son environnement technique, afin d’agréger différents flux de communication temps réel du mandant.

# Le protocole Matrix

## Matrix qu’est-ce que c’est ?

Matrix est un protocole de communication ouvert et décentralisé pour la communication sécurisée en temps réel sur internet. Il fournit une base pour les services de messageries et de VoIP et pour l’internet des objets. Il est Open Source, ce qui lui permet d’éviter les licences propriétaires et est conçu pour être interopérable. Il permet également d’intégrer d’autres applications de messagerie tels que Telegram, Discord, WhatsApp, etc.

Pour accéder à Matrix, il est possible de se connecter sur des serveurs publics ou de créer son propre serveur. On y accède ensuite via un client qui utilise ce protocole comme FluffyChat, Element, Cinny, etc.

## Matrix – Le fondamental

Le protocole Matrix est conçu pour être une infrastructure de communication décentralisée, sécurisée et interopérable. Voici les principaux concepts qui guident sa conception :

**Fédération :** Le protocole Matrix repose sur le principe de la fédération, ce qui signifie que différents serveurs Matrix peuvent interagir et communiquer les uns avec les autres. Cela permet donc aux utilisateurs de créer des comptes sur n’importe quel serveur compatible Matrix et de communiquer avec des utilisateurs sur d’autres serveur. Cela représente le fonctionnement des mails à travers différents fournisseurs.

**Décentralisation :** L'un des principaux principes de conception du protocole Matrix est la décentralisation. Contrairement aux services de messagerie centralisés, où un seul fournisseur contrôle l'ensemble du système, Matrix repose sur de nombreux serveurs indépendants (homeservers). Chaque serveur héberge des comptes d'utilisateurs et des données, garantissant que le contrôle des données est distribué.

**Identifiants Uniques :** Le protocole utilise des identifiants uniques pour chaque utilisateur et chaque pièce. Les IDs Matrix sont utilisés pour localiser et identifier les ressources sur le réseau.

**Pièces (Rooms) :** Le concept de pièces est au cœur du protocole Matrix. Les utilisateurs communiquent dans des pièces, qui peuvent être publiques ou privées. Les pièces peuvent être organisées autour de thèmes spécifiques, de groupes d'utilisateurs ou de projets.

**Messages et Événements :** Le protocole traite les messages sous forme d'événements. Chaque action (envoi de message, ajout d'utilisateur, changement d'autorisation, etc.) est représentée comme un événement dans l'historique de la pièce.

**Synchronisation Continue :** Les clients et les serveurs travaillent ensemble pour garantir que les clients sont constamment synchronisés avec les dernières mises à jour dans les pièces. Cela assure une expérience de communication fluide et en temps réel.

**Chiffrement de Bout en Bout :** La sécurité et la confidentialité sont prioritaires dans la conception du protocole. Le chiffrement de bout en bout permet de sécuriser les messages de telle sorte que seuls les destinataires légitimes puissent les lire.

**Extensions et Intégrations :** Le protocole Matrix est conçu pour être extensible. Cela signifie que de nouvelles fonctionnalités peuvent être ajoutées et que des intégrations avec d'autres systèmes sont possibles.

**Règles de Communication :** Le protocole définit les règles et les normes de communication entre les clients et les serveurs. Il spécifie comment les messages sont envoyés, reçus, stockés et synchronisés.

**Transparence :** L'architecture et le code du protocole sont ouverts, ce qui signifie que la conception et le fonctionnement sont transparents et accessibles à tous. Cela encourage la confiance et la participation de la communauté.

## Les rooms

Il est possible de créer des pièces (rooms) qui est un espace de discussion où les utilisateurs peuvent échanger des messages, des fichiers et d'autres contenus en temps réel. Les utilisateurs peuvent participer à plusieurs pièces à la fois.

**Création de Rooms :** Les utilisateurs ont la possibilité de créer leurs propre rooms. Lors de la création d'une room, l'utilisateur peut définir un nom, une description et d'autres paramètres pour définir le contexte de la conversation.

**Types de Rooms :** Les rooms peuvent être de différents types en fonction de leur accessibilité et de leurs règles de participation. Il existe principalement deux types de rooms

**Rooms Publiques :** Les rooms publiques sont accessibles à tous les utilisateurs de Matrix. Les utilisateurs peuvent rejoindre ces rooms sans invitation et participer aux discussions.

**Rooms Privées :** Les rooms privées sont restreintes et nécessitent une invitation pour y accéder. Les créateurs de la room peuvent inviter des utilisateurs spécifiques et contrôler qui peut rejoindre la conversation.

**Participation et Interaction :** Une fois dans une room, les participants peuvent échanger des messages en temps réel. Les messages peuvent contenir du texte, des images, des liens, des fichiers et d'autres contenus multimédias.

**Interopérabilité :** Une caractéristique importante des rooms Matrix est leur capacité à être interopérables entre différents serveurs Matrix. Cela signifie que les utilisateurs sur des serveurs différents peuvent participer à la même room et communiquer ensemble, tout comme des adresses e-mail peuvent communiquer entre différents fournisseurs.

**Historique des Messages :** Les rooms conservent un historique des messages échangés. Cela signifie que lorsque les utilisateurs rejoignent une room, ils peuvent consulter les messages précédents pour se mettre à jour sur la conversation.

**Gestion et Modération :** Les créateurs et les administrateurs d'une room peuvent généralement gérer la configuration de la room, inviter de nouveaux participants, supprimer des messages indésirables et gérer d'autres aspects de la conversation.

## Les bridges

Matrix prend en charge l'acheminement des messages de différentes applications de discussion vers Matrix. Ces ponts (bridge) sont des programmes qui permettent de relier Matrix à d'autres protocoles de messagerie. Cela permet aux utilisateurs de ces services externes de communiquer avec des utilisateurs Matrix sans quitter leur application préférée. Voici comment ça fonctionne pour un pont Matrix-Telegram :

**Configuration du Pont :** Sur un serveur Matrix, un administrateur configure le bridge spécifique. Ce bridge est un logiciel qui sera responsable de la communication entre les deux plateformes. Le bridge Matrix-Telegram est configuré pour se connecter aux serveurs de Telegram grâce à l’API de Telegram.

**Communication avec Telegram :** Lorsqu'un utilisateur de Telegram envoie un message dans une conversation Matrix, le bridge reçoit ce message depuis l'API Telegram et il interprète le message et le traduit dans un format compréhensible par le protocole Matrix.

**Synchronisation des Messages :** Le bridge envoie ensuite ce message traduit vers le serveur Matrix auquel il est connecté. Le serveur Matrix distribue alors ce message à la room auquel les utilisateurs Matrix et le participant de Telegram communiquent.

**Envoi vers Matrix :** Lorsqu’un utilisateur de Matrix envoie un message dans une communication Telegram, le processus est le même.

L’avantage principal des bridges est la transparence pour les utilisateurs. Cela permet de faire croire aux utilisateurs Matrix et aux utilisateurs Telegram qu’ils sont dans la même room.

## Les spaces

Les espaces (spaces) sont une extension du protocole Matrix qui permet d’organiser et de structurer les rooms de manière hiérarchique. Cela permet une meilleure organisation des conversations.

**Création de Spaces :** Les utilisateurs peuvent créer des spaces pour regrouper plusieurs rooms en fonction de leurs besoins. Un space peut être créé autour d'un thème spécifique, d'un projet en cours, d'un groupe d'utilisateurs ou de toute autre organisation logique.

**Hiérarchie et Structuration :** À l'intérieur d'un space, nous pouvons créer des catégories et organiser les rooms de manière hiérarchique. Les catégories aident à structurer les discussions en groupant des rooms similaires ou liées sous un même en-tête.

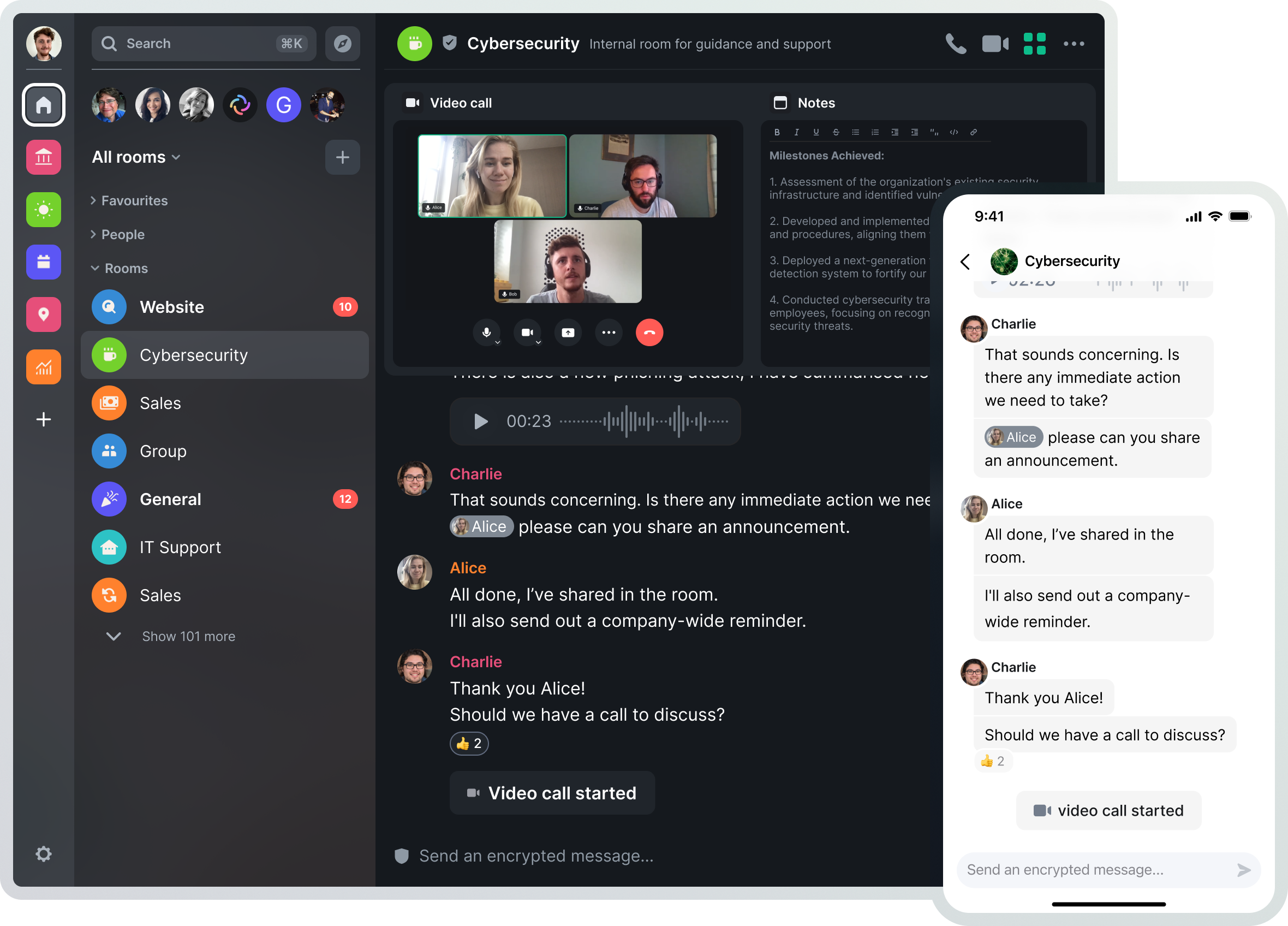
**Navigation Facilitée :** Les spaces facilitent la navigation entre les différentes rooms. Plutôt que d'avoir une liste de toutes les rooms auxquelles nous participons, les spaces nous permettent de visualiser les rooms dans un contexte organisé et compréhensible.

**Thématiques Cohérentes :** Les spaces offrent un moyen de rassembler des conversations ayant des liens thématiques sous une même bannière. Par exemple, un space pourrait regrouper toutes les rooms liées à un projet de développement, un espace personnel ou une communauté spécifique.

**Contrôle d'Accès :** Les propriétaires d'un space peuvent définir des règles d'accès pour les rooms qui en font partie. Cela permet de contrôler qui peut rejoindre les rooms et participer aux discussions.

**Gestion Centralisée :** En tant qu'administrateur d'un space, il est possible de gérer les paramètres du space et les paramètres individuels des rooms qu'il contient. Cela permet une gestion centralisée de l'ensemble des rooms liées à un contexte spécifique.

**Gestion des Conversations :** Les spaces aident à garder les discussions organisées en évitant que les utilisateurs se perdent parmi un grand nombre de rooms indépendantes.



Ci-dessus, une capture d’écran du client Element. Les icones à gauche sont des spaces. Ensuite, nous pouvons voir les différentes rooms/bridges de l’utilisateur. Au milieu se trouve une visioconférence en cours et dessous la conversation de la room et à droite la version téléphone.

## Les APIs

Les API du protocole Matrix fournissent des points d'entrée pour que les développeurs puissent créer des applications, des clients personnalisés, des intégrations et des fonctionnalités qui interagissent avec le protocole et les serveurs Matrix. Voici quelques-unes des utilisations principales des API du protocole Matrix :

**Clients Matrix personnalisés :** Les API permettent aux développeurs de créer des clients Matrix personnalisés. Ces clients peuvent avoir des interfaces utilisateur spécifiques, des fonctionnalités uniques et des expériences utilisateur adaptées aux besoins.

**Intégrations :** Les API offrent la possibilité d'intégrer Matrix dans d'autres applications ou services. Cela peut inclure l'ajout de fonctionnalités de discussion en temps réel à des plateformes existantes ou la création de passerelles pour relier Matrix à d'autres protocoles de messagerie.

**Bridges :** Les API permettent la création de bridges entre Matrix et d'autres protocoles de messagerie.

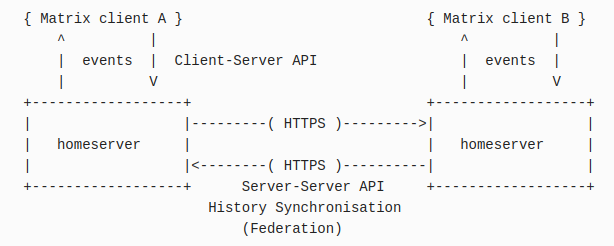
**Automatisation :** Les API permettent de créer des bots et des scripts automatisés qui effectuent des actions spécifiques, comme l'envoi de messages programmés, la gestion des invitations dans les pièces, etc.

**Développement d'applications personnalisées :** Les développeurs peuvent utiliser les API pour créer des applications personnalisées qui utilisent les fonctionnalités du protocole Matrix. Cela peut inclure des applications de gestion, des outils de collaboration, etc.

**Analyse de Données :** Les API permettent d'extraire des données et des statistiques à partir du réseau Matrix. Cela peut aider à comprendre les modèles de communication, à surveiller l'activité des utilisateurs, etc.

**Chiffrement de Bout en Bout :** Les API permettent d'implémenter le chiffrement de bout en bout dans les clients et les applications. Cela assure la sécurité des messages en les chiffrant localement avant l'envoi et en les déchiffrant localement à la réception.

## Principe de communication



Ci-dessus, nous pouvons observer un échange d’événements entre deux clients Matrix qui ne sont pas sur le même serveur.

Par exemple, pour que le client A envoie un message au client B, le client A effectue une requête d’ajout d’un événement (HTTP PUT), dans ce cas un message, au homeserver (serveur où l’utilisateur est enregistré) à l'aide de l'API client-serveur. Le homeserver de A ajoute cet événement à sa copie du graphe d'événements de la salle, en signant le message dans le contexte du graphe pour en garantir l'intégrité.

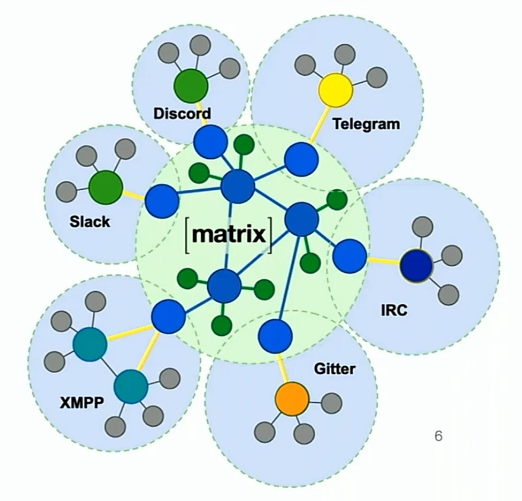
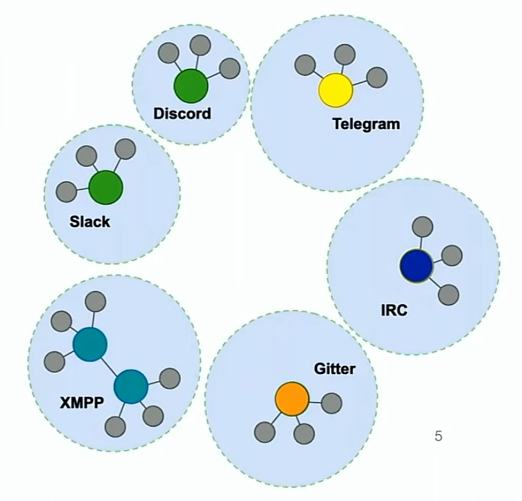
L'Event Graph est une représentation sous forme de graphe qui enregistre l'historique des événements au sein d'une room dans le protocole Matrix. Il assure la cohérence, la synchronisation et l'ordre des interactions et des communications entre les utilisateurs et les serveurs dans le réseau Matrix.

Puis, le homeserver de A réplique ensuite le message au HomerServer de B en effectuant la même requête (HTTP PUT) à l'aide de l'API serveur-serveur. Le homeserver de B authentifie la demande, valide la signature de l'événement, autorise le contenu de l'événement et l'ajoute à sa copie du graphe d'événements de la salle. Le client B reçoit ensuite le message de son homeserver en effectuant une requête de demande au serveur (HTTP GET) pour savoir si des nouveaux événements sont arrivés.

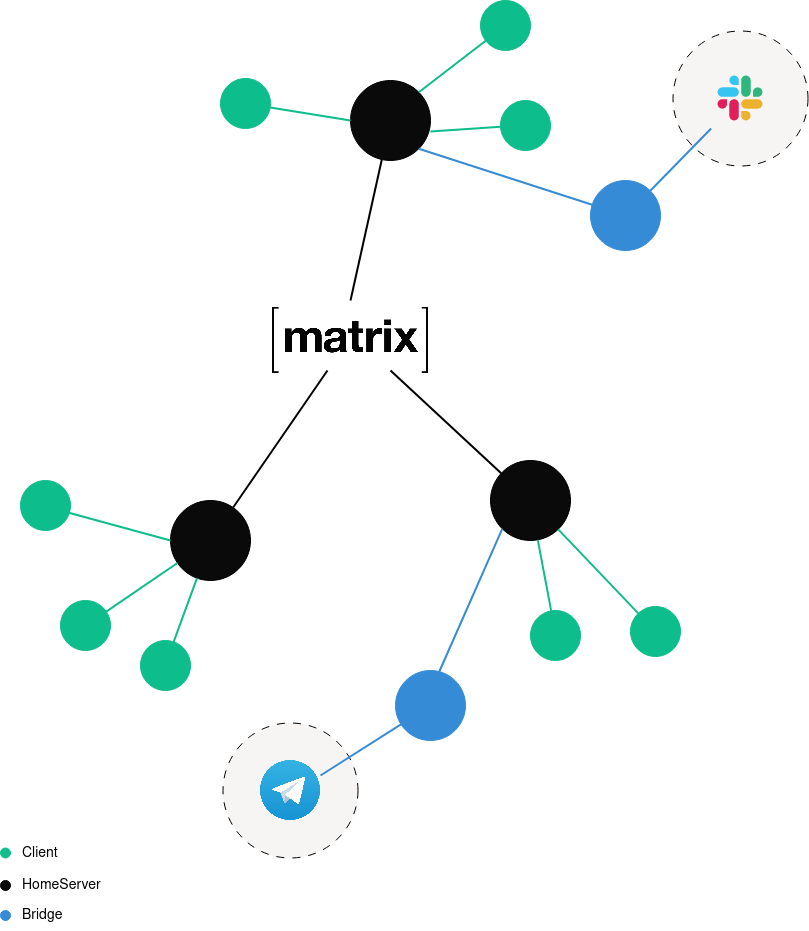
# Matrix – Architecture

Comme cité ci-dessus, l'architecture de Matrix est conçue pour être décentralisée, interopérable et sécurisée. Elle repose sur une combinaison de serveurs, de clients et de protocoles pour permettre des communications en temps réel.

Dans un contexte plus complexe nous pouvons observer que Matrix permet de regrouper différents systèmes entre eux.



Si nous faisons un zoom sur l’architecture de Matrix :



Les homeservers sont les serveurs Matrix qui contiennent les utilisateurs, les rooms, etc. Nous observons la fédération des serveurs grâce au protocole Matrix. Les clients accèdent au serveur afin d’afficher la vue pour les utilisateurs. Le homeserver est interopérable avec Telegram et Slack (pour ce schéma) grâce aux bridges qui assurent la communication entre les deux systèmes.

## Les Acteurs

Les principaux acteurs sont les :

**Clients :** Les clients Matrix sont des applications utilisées par les utilisateurs pour accéder au réseau. Ils peuvent être des applications de bureau, des applications mobiles ou même des clients basés sur le Web. Les clients permettent aux utilisateurs de se connecter aux serveurs, de rejoindre des pièces, d'envoyer et de recevoir des messages, et d'interagir en temps réel.

**Serveurs Matrix (Homeservers) :** Les serveurs Matrix sont des composants centraux de l'architecture. Chaque utilisateur est lié à un serveur Matrix, qui gère les comptes d'utilisateurs, les messages, les pièces et d'autres données associées à cet utilisateur. Les serveurs Matrix communiquent entre eux via la fédération pour permettre l'interopérabilité.

**Autres plateformes :** Grâce au bridge, des plateformes de messagerie différentes (comme Slack, IRC, Discord, etc.) peuvent communiquer avec le réseau Matrix. Cela élargit l'interopérabilité du protocole.

# Serveur

## Environnement

L’installation est sur une machine virtuelle hébergé sur les serveurs de l’EPFL. La VM est sur le système d’exploitation Linux avec une version Ubuntu22.04. Elle a une capacité de 20GB et une installation de base avec ssh d’installé. Elle est donc accessible à distance avec ssh.

## Synapse

Il existe actuellement quatre serveurs Open Source conçus pour le protocole Matrix : Synapse, Construct, Conduit et Dendrite

Synapse est actuellement le serveur le plus utilisé et le seul à proposer une version stable. C’est celui-ci qui va être installé.

## Serveur d’identification

Sydent est un serveur d'identité pour le protocole Matrix. Il permet aux utilisateurs de Matrix de prouver qu'ils possèdent une adresse e-mail ou un numéro de téléphone, et permet à d'autres utilisateurs de Matrix de les rechercher en utilisant cette adresse e-mail ou ce numéro de téléphone.

Il n’est pas nécessaire d’installer un serveur d’identification lorsqu’on installe son homeserver car il est possible d’utiliser celui de Matrix.org.

Cependant, si nous voulons permettre la recherche d'utilisateurs via des adresses e-mail et des numéros de téléphone dans une fédération privée de plusieurs homeserver, Sydent pourrait être utile.

# Installation sans automatisation

Dans un premier temps, l’installation est faite sans automatisation : sans docker compose ou sans docker file. La raison de ce choix est liée aux différents problématiques cités plus bas dans le rapport.

Le serveur de Matrix (Synapse) a besoin de différents composants pour fonctionner :

* Le serveur qui utilise le protocole Matrix, Synapse
* Une base de données : Postgres
* Un proxy : nginx

L’installation suit la procédure proposée par Matrix : <https://matrix-org.github.io/synapse/latest/setup/installation.html>

## Synapse

Cette commande exécute un conteneur Docker basé sur l'image matrixdotorg/synapse:latest, en montant un volume nommé synapse-data. Elle définit plusieurs variables d'environnement pour configurer le serveur Synapse (le nom du serveur et l’envoie de statistique). Elle permet d’avoir une configuration de base pour la suite

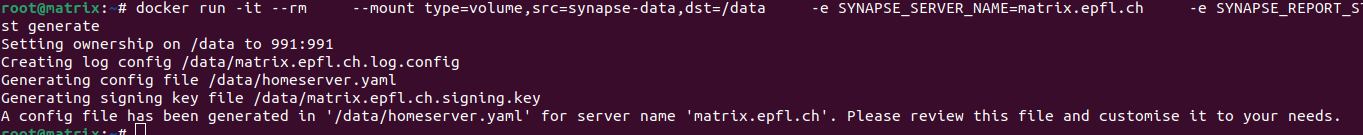
docker run -it --rm \

--mount type=volume,src=synapse-data,dst=/data \

-e SYNAPSE\_SERVER\_NAME=matrix.epfl.ch \

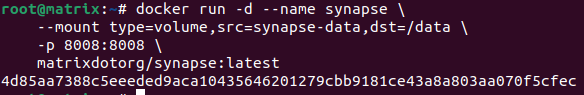
-e SYNAPSE\_REPORT\_STATS=yes \

matrixdotorg/synapse:latest generate



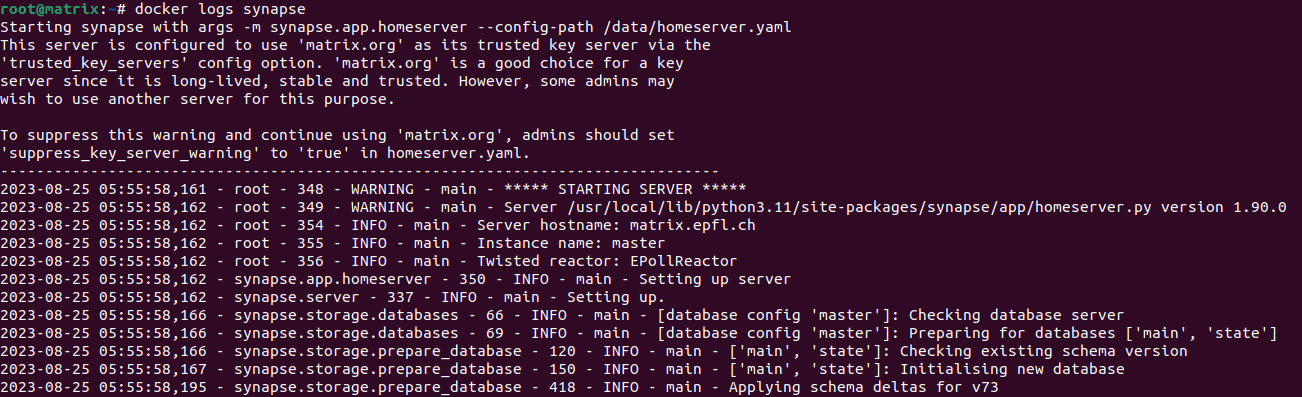
Cette commande exécute le conteneur Docker synapse en se basant sur la configuration créée ci-dessus. Elle permet également de mapper un port de l'hôte (8008) sur un port du conteneur (8008) ce qui redirige les requêtes réseau de l’hote vers le docker.

docker run -d --name synapse \  
 --mount type=volume,src=synapse-data,dst=/data \  
 -p 8008:8008 \  
 matrixdotorg/synapse:latest

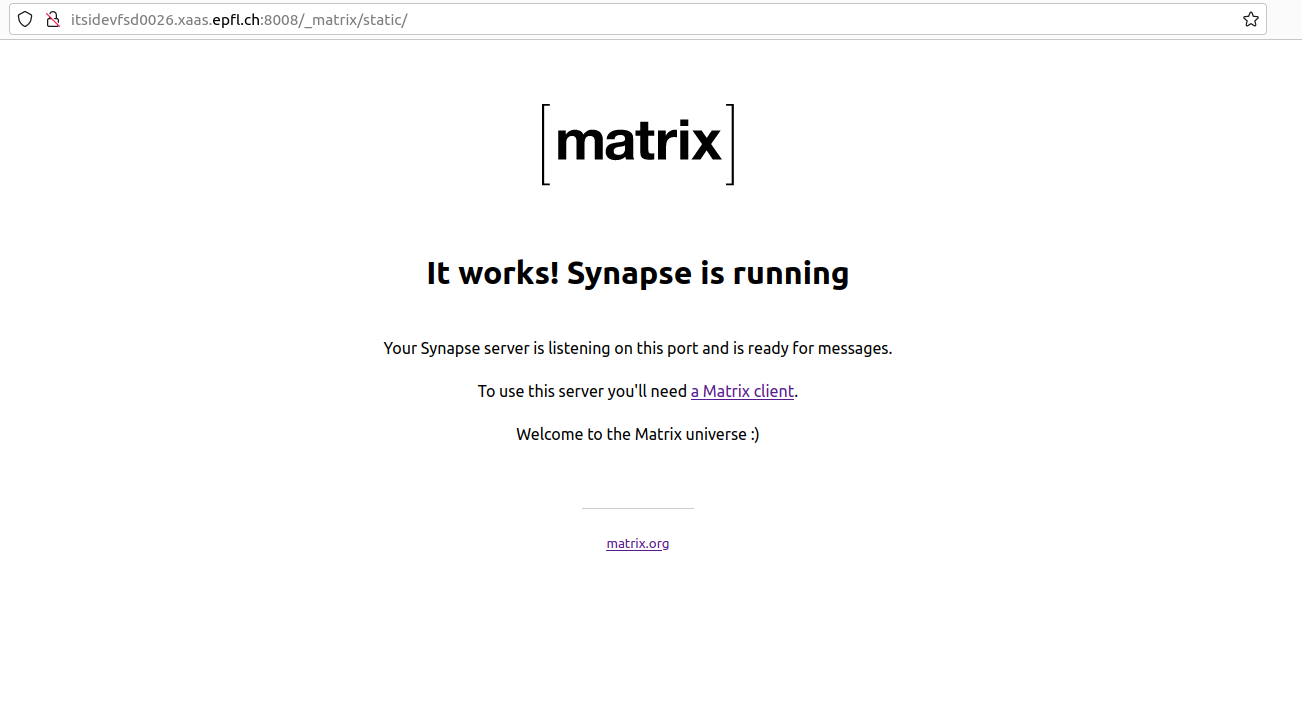


Pour vérifier que le serveur à démarrer sans problème, il est possible d’aller voir les logs :

docker logs synapse



Comme le serveur a démarré correctement, on peut y accéder depuis l’adresse de la machine avec le port 8008.

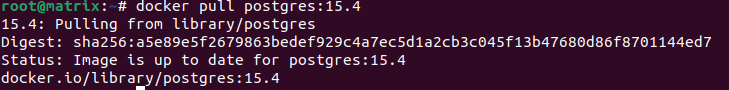


## PostgreSQL

Synapse a besoin d’une base de données pour y stocker les utilisateurs, les rooms, les conversations, etc. Le choix s’est tourné sur PostgreSQL car c’est un logiciel OpenSource et c’est la base de données qui est utilisés par Matrix.

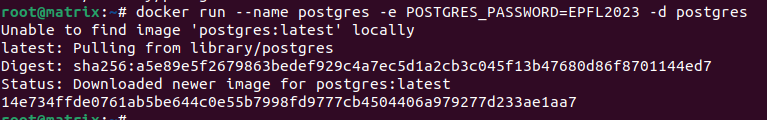
Téléchargement de l’image PostgreSQL version 15.4.

docker pull postgres:15.4



Cette commande exécute l'image PostgreSQL, monte le volume postgres\_data pour stocker les données de PostgreSQL de manière persistante, elle configure le mot de passe de l'utilisateur superutilisateur, et lance le conteneur en arrière-plan.

docker run --name postgres --mount type=volume,source=postgres\_data,target=/var/lib/postgresql/data -e POSTGRES\_PASSWORD=EPFL2023 -d postgres



Une fois l’image démarré, il faut créer la base de données que Synapse utilisera. Pour ce faire, il faut se connecter au Docker en ligne de commande.

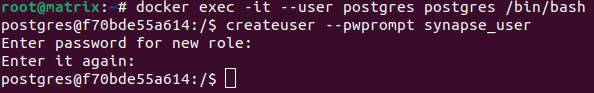
docker exec -it --user postgres postgres /bin/bash



Création d’un utilisateur qui sera propriétaire de la base de données

createuser --pwprompt synapse\_user

### Password : EPFL2023



Il est important de spécifier l’encodage UTF8 lors de la création de la base de données.

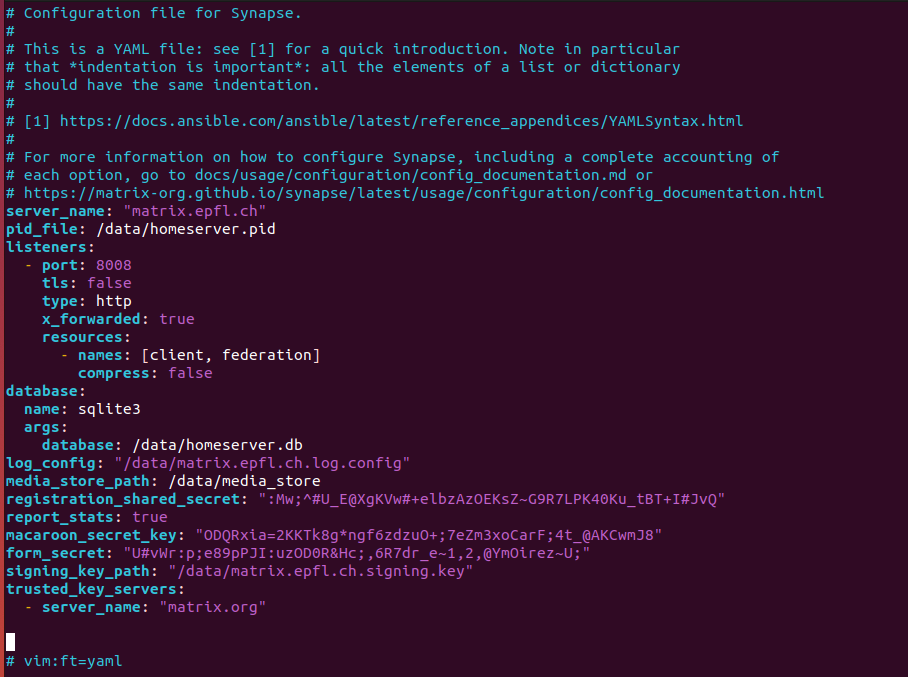
createdb --encoding=UTF8 --locale=C --template=template0 --owner=synapse\_user synapse

Quitter le Docker postgres

Maintenant que la base de données est créée, il faut la spécifier à synapse.

vi /var/lib/docker/volumes/synapse-data/\_data/homeserver.yaml

Voici le fichier de configuration par défaut. Il faut remplacer la section database.



Les informations fournies permettent de configurer également les charges pour le serveur PostgreSQL (mis par défaut). Le nom de l’host est l’ID du Docker de PostgreSQL car il n’y a pas de DNS configuré.

database:

name: psycopg2

args:

user: synapse\_user

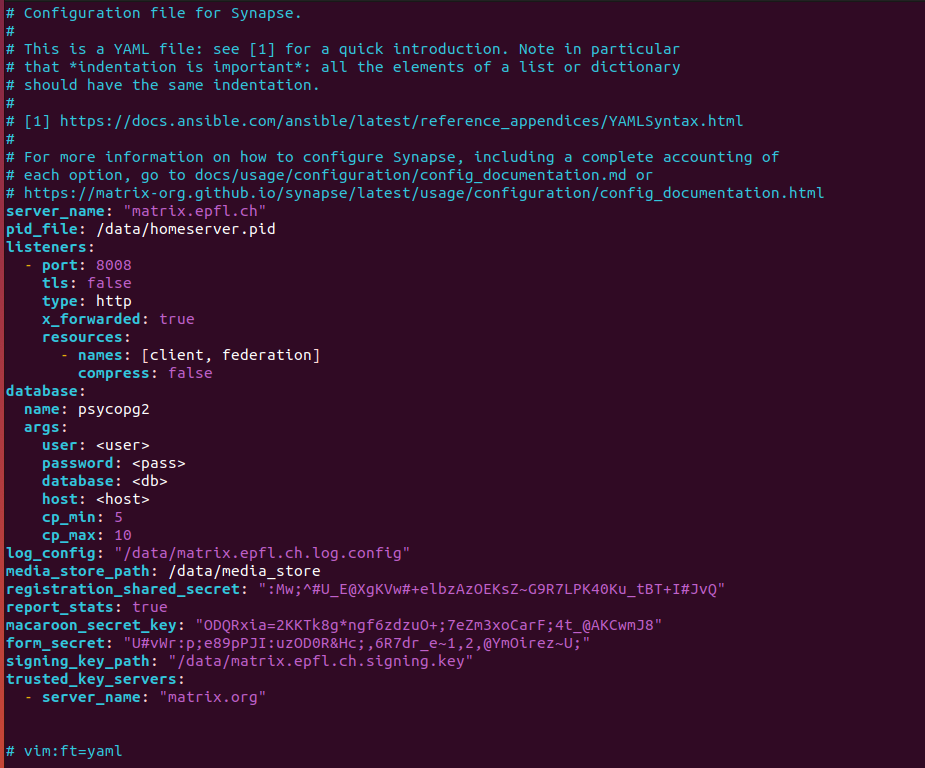
password: EPFL2023

database: synapse

host: 0138c269487b

cp\_min: 5

cp\_max: 10



## Reverse Proxy

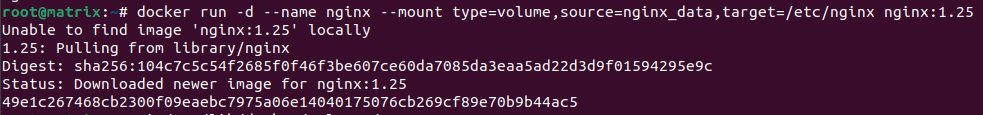
Matrix a besoin d’un reverse proxy pour faire l’intermédiaire entre les utilisateurs et le serveur. Il va également permettre de vérifier si un certificat SSL est installé, de faire du load balancing, mise en cache, etc.

Exécution de l’image Nginx en montant le volume où seront stocker les fichiers.

docker run -d --name nginx --mount type=volume,source=nginx\_data,target=/etc/nginx nginx:1.25

Execution du serveur en liant les ports 80 de l’hôte et du conteneur Docker.

docker run --name nginx -p 80:80 -d nginx



Ensuite, il va falloir modifier les données du fichier de configuration. Les informations ci-dessous reprennent la configuration de base fourni par Matrix.

vi /var/lib/docker/volumes/nginx\_data/\_data/conf.d/default.conf

Code:

server {

listen 443 ssl http2;

listen [::]:443 ssl http2;

# For the federation port

listen 8448 ssl http2 default\_server;

listen [::]:8448 ssl http2 default\_server;

server\_name proxy.matrix.epfl.ch;

location ~ ^(/\_matrix|/\_synapse/client) {

# note: do not add a path (even a single /) after the port in `proxy\_pass`,

# otherwise nginx will canonicalise the URI and cause signature verification

# errors.

proxy\_pass http://localhost:8008;

proxy\_set\_header X-Forwarded-For $remote\_addr;

proxy\_set\_header X-Forwarded-Proto $scheme;

proxy\_set\_header Host $host;

# Nginx by default only allows file uploads up to 1M in size

# Increase client\_max\_body\_size to match max\_upload\_size defined in homeserver.yaml

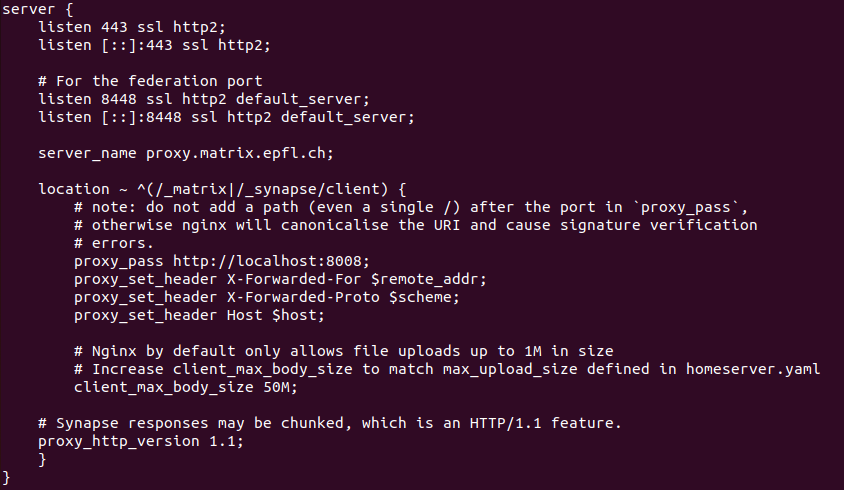
client\_max\_body\_size 50M;

# Synapse responses may be chunked, which is an HTTP/1.1 feature.

proxy\_http\_version 1.1;

}

}



## Certificat SSL

Pour pouvoir accéder au serveur depuis le client, synapse doit avoir un certificat SSL. OpenSSL a été utilisé pour en créer un.

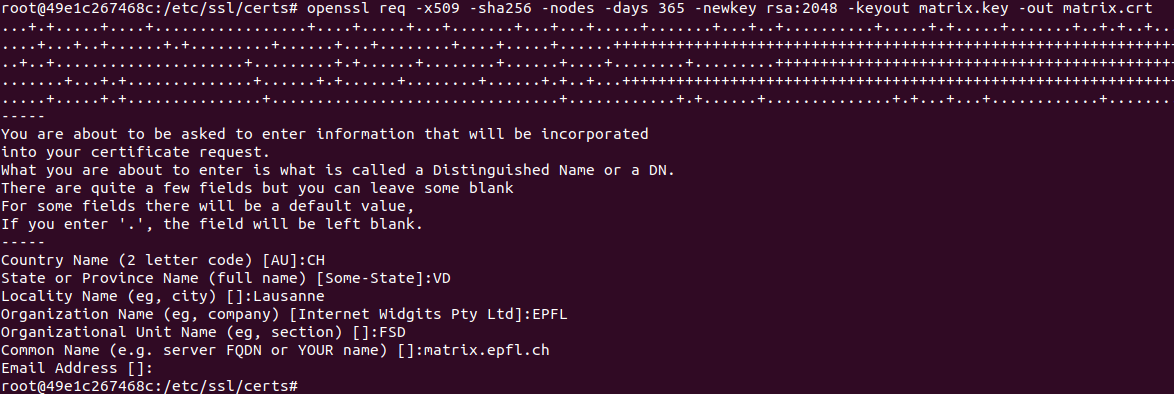
Accès au serveur en ligne de commande :

docker exec -it synapse /bin/bash

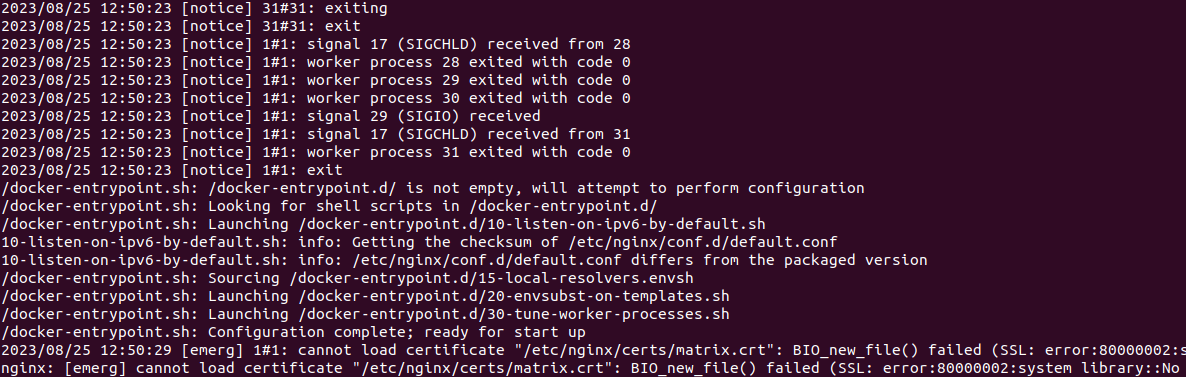
Création d’un certificat avec une clé autogénéré :

openssl req -new- x509 -nodes -days 365 -key key.pem > cert.pem

req -x509 -sha256 -nodes -days 365 -newkey rsa:2048 -keyout matrix.key -out matrix.crt



Afin d’éviter l’erreur ci-dessous, il faut aller modifier le chemin d’accès du certificat et de la clé.



Accès au fichier de configuration du serveur Nginx.

vi /var/lib/docker/volumes/nginx\_data/\_data/conf.d/default.conf

Remplacement des chemins d’accès :



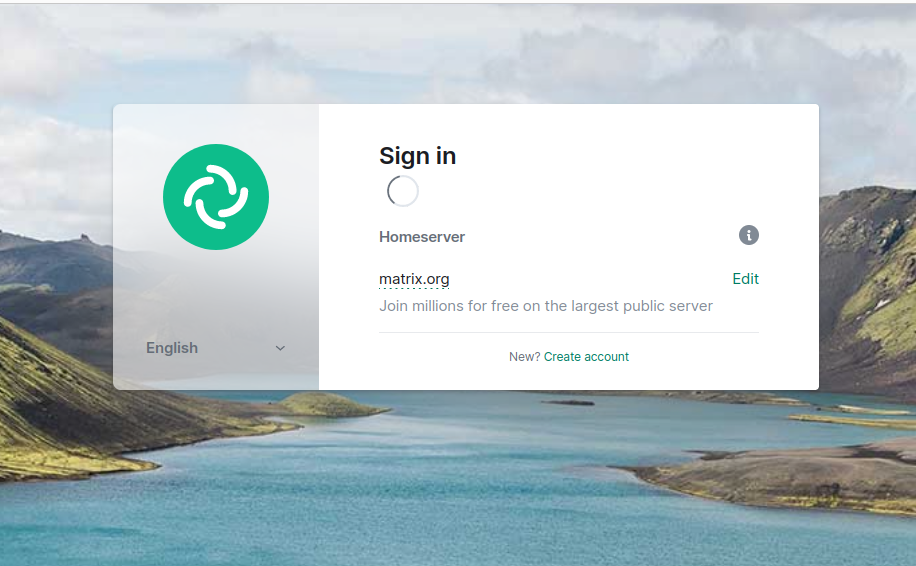
## Test avec le client

Les tests se feront avec le client Element. Il est le client le plus largement utilisé.

Execution de l’image Element en liant le port 80 de l’hôte et du Docker.

docker run -p 80:80 vectorim/element-web

On accède ensuite à la page de connexion via la page par défaut de localhost.



Le serveur par défaut est celui de Matrix. Il faut cliquer sur Edit pour pouvoir renseigner notre propre serveur.



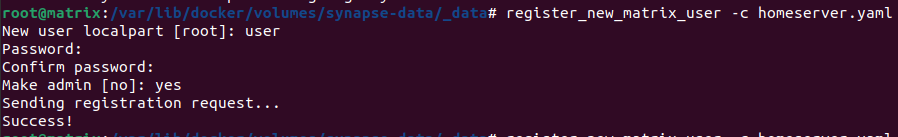
Comme aucun utilisateur existe, il faut en créer des nouveaux pour pouvoir se connecter.

Pour se faire, se rendre dans le dossier de synapse :

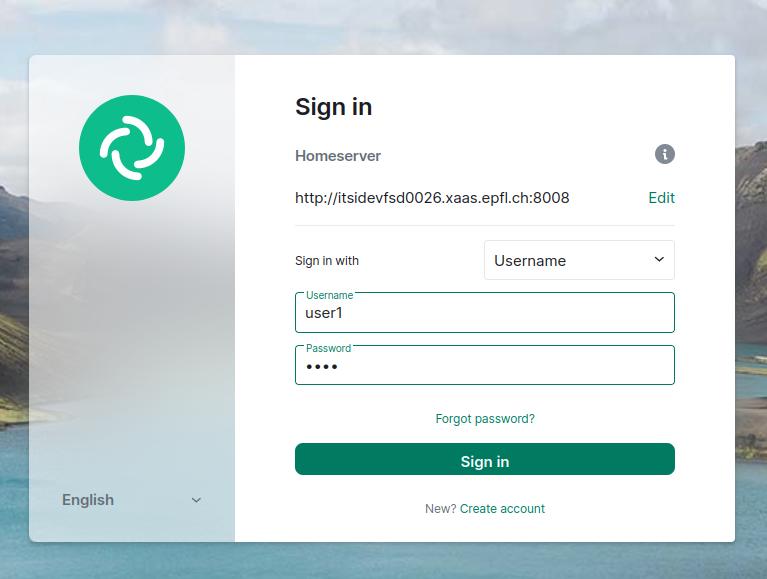
cd /var/lib/docker/volumes/synapse-data/\_data/

Exécuter la commande pour créer un nouvel utilisateur et renseigner ses informations :

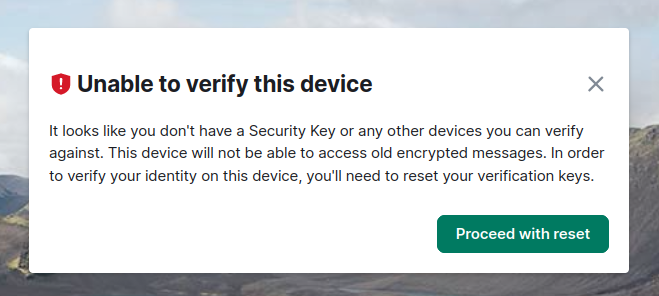
register\_new\_matrix\_user -c homeserver.yaml



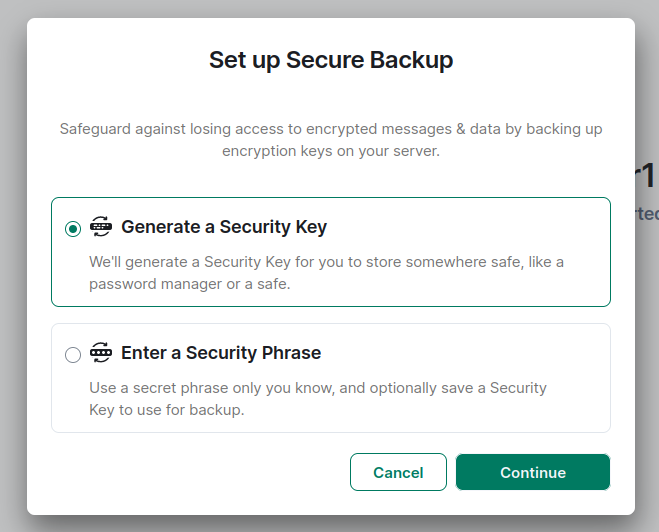
Se connecter avec les informations de l’utilisateur :

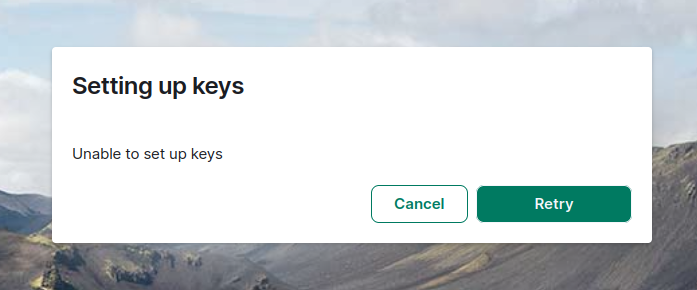


Lors de la première connexion, il demande d’avoir une clé de sécurité pour accéder à nos conversations en cas de perte d’accès aux données crypté ou en cas de backup.

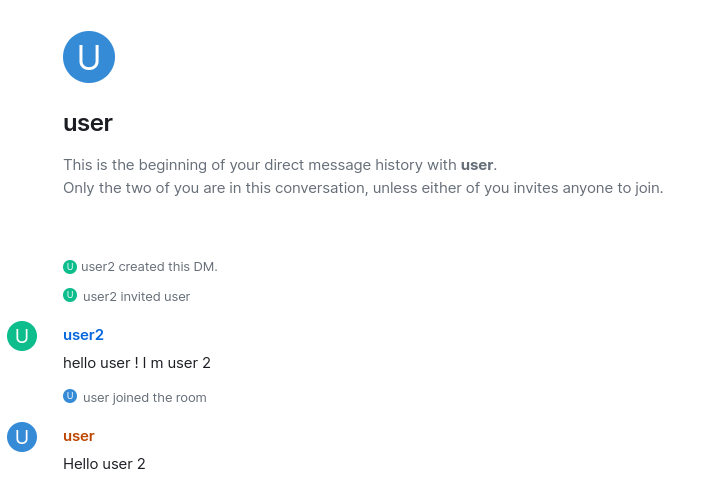


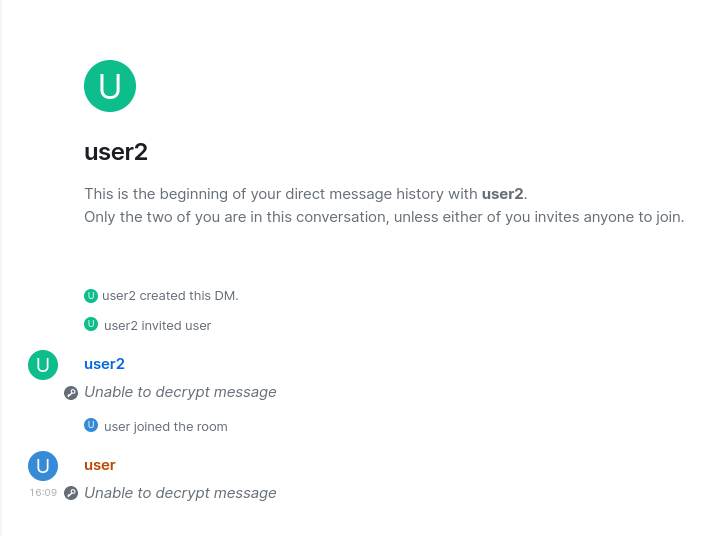
Choisir une option appropriée au contexte :





Test d’envois de message entre deux utilisateurs :



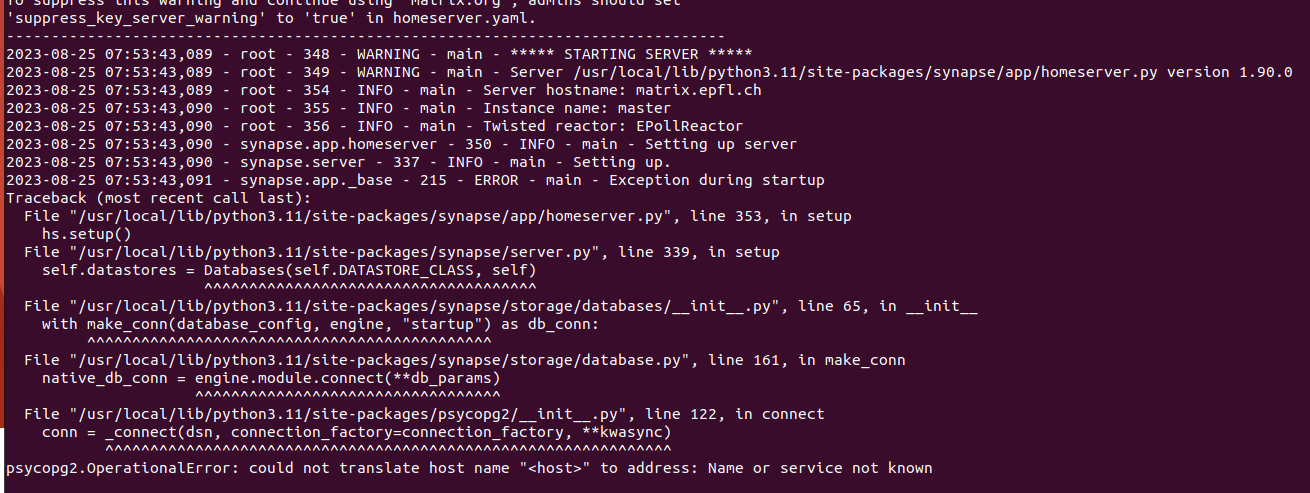


# Problèmes rencontrés

## Nom d’hôte

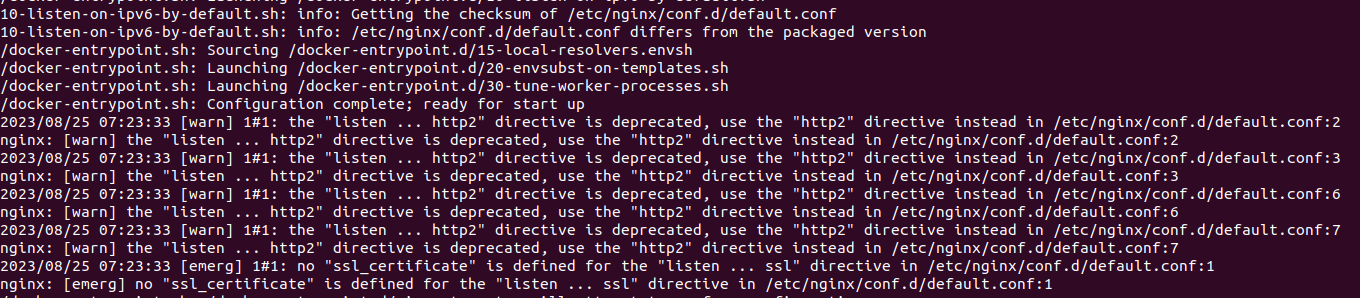
Lors de la première utilisation de Docker, le nom d'hôte spécifié était incorrect. Après avoir effectué plusieurs recherches sur Docker, il est devenu évident qu'il fallait renseigner l'ID du conteneur Docker.

psycopg2.OperationalError: could not translate host name "<host>" to address: Name or service not known

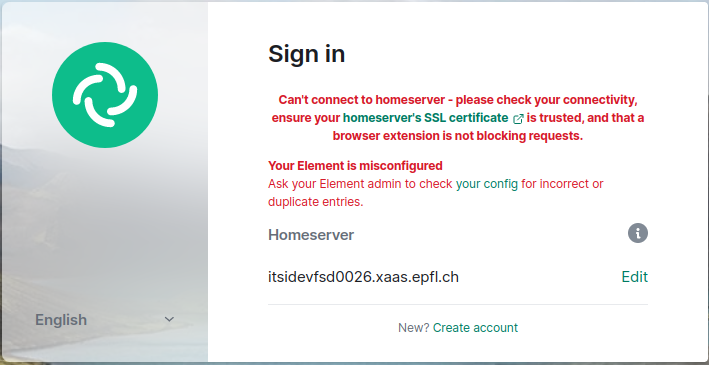


## Certificat SSL

Une fois l’installation du serveur Nginx faite, le serveur s’arrêtait directement après son lancement. On observe que c’était à cause du certificat SSL. Dans le fichier de configuration de Nginx, tout ce qui concerne le SSL a été mis en commentaire, y compris le port, le chemin d'accès, etc.



Cependant, lors de l'accès au serveur depuis le client, une erreur est survenue. La création du certificat SSL cité précédemment a donc été réalisé suite à cette situation.

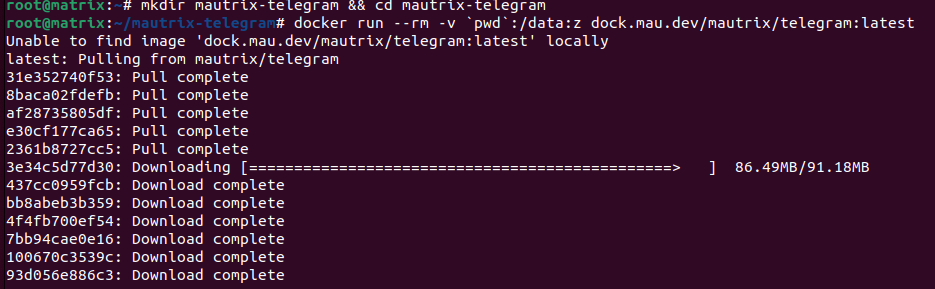


# Bridges

Telegram

mkdir mautrix-telegram && cd mautrix-telegram

docker run --rm -v `pwd`:/data:z dock.mau.dev/mautrix/telegram:latest



# Glossaire

**Room :** Salle de conférence sur MATRIX. Les pièces peuvent être « publiques » (c’est-à-dire que n’importe qui peut entrer librement dans la pièce) ou « privée » (c’est-à-dire qu’on ne peut pas entrer dans la pièce avant d’y être invité).

**Space :** Une liste ou un groupe de room.

**Bridge :** Un pont de communication qui permet la communication entre une salle MATRIX et un autre réseau de communication tel que Discord ou Telegram.

**Chiffrement de Bout en Bout :** Une fonctionnalité de sécurité qui garantit que les messages sont chiffrés sur l’appareil de l’expéditeur et ne peuvent être déchiffrés que par le destinataire prévu.