Matrix / Element

Matrix

> Cette documentation n'est pas complète. Quelques éléments sont en cours de rédaction. Néanmoins, des éléments essentiels devraient pouvoir vous aider dans votre déploiement de Matrix, raison pour laquelle nous avons choisi de la publier.

Cette documentation décrit l'installation d'un serveur Synapse (actuellement le seul stable), d'un client Element, et de services annexes : synapse-admin pour une interface graphique et MAS pour pouvoir s'authentifier en SSO sur le client Android Element X.

> La variable allow_guest_access: true crée des utilisateurs invités dans la base de données avec des noms tels que @1:exemple.fr, puis @2:exemple.fr, etc. à chaque erreur de connexion avec LDAP/SSO (et d'autres problèmes).

Création de la VM Debian

> Dans cette documentation, la VM a besoin de 4 noms de domaines : → pour le serveur Matrix Synapse (**matrix.exemple.fr**); → pour le client Element (**element.exemple.fr**) ; → pour l'interface web d'administration Synapse Admin (**admin.matrix.exemple.fr**) ; → pour le service MAS (**auth.matrix.exemple.fr**).

Création de la **VM** : prévoir 4 CPU et 4 Go RAM.

Installation du serveur Matrix Synapse

> Pour l'installation, je me suis aidée : → du README disponible sur le GitHub → de la documentation de Element

Installer le paquet debian matrix-synapse-py3

Dans notre cas, on utilisera exemple.fr comme nom de serveur, avec srv-matrix.exemple.fr comme machine. Il faut donc ajouter un enregistrement DNS pour faire une redirection des flux matrix qui

arrive sur exemple.fr vers srv-matrix.exemple.fr :

```
_matrix-fed._tcp.exemple.fr. 3600    IN      SRV     10 0 443 srv-matrix.exemple.fr.
```

Lorsque cela est demandé, mettre **exemple.fr**, puis **No**. Cela va créer des fichiers de configuration dans le dossier *conf.d* de Matrix Synapse (server_name.yaml et report_stats.yaml)

```
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
```

Installer et configurer PostgreSQL

```
# Import the repository signing key:
sudo apt install curl ca-certificates
sudo install -d /usr/share/postgresql-common/pgdg
sudo curl -o /usr/share/postgresql-common/pgdg/apt.postgresql.org.asc --fail
https://www.postgresql.org/media/keys/ACCC4CF8.asc
# Create the repository configuration file:
sudo sh -c 'echo "deb [signed-by=/usr/share/postgresql-common/pgdg/apt.postgresql.org.asc]
https://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" >
/etc/apt/sources.list.d/pgdg.list'
sudo apt update
sudo apt -y install postgresql
```

Synapse a besoin de la bibliothèque client python postgres pour se connecter à une base de données postgres. Donc on **installe libpq5**.

```
sudo apt install libpq5
```

Créer un utilisateur **matrix-synapse** et une base de données **synapse** :

```
sudo -u postgres bash
createuser matrix-synapse
createdb --encoding=UTF8 --locale=C --template=template0 --owner=matrix-synapse synapse
```

Créer le fichier de configuration `/etc/matrix-synapse/conf.d/database.yaml` :

```
database:
  name: psycopg2
  args:
    user: matrix-synapse
    dbname: synapse
    host: /run/postgresql
    cp_min: 5
    cp_max: 10
```

Configurer Matrix Synapse

Créer le fichier `/etc/matrix-synapse/conf.d/synapse.yaml` avec la configuration suivante (pour éviter de toucher au fichier `/etc/matrix-synapse/homeserver.yaml`) :

```
# For more information, see https://element-
hq.github.io/synapse/latest/usage/configuration/config_documentation.html

public_baseurl: https://matrix.exemple.fr

# Desactiver la possibilite de changer son nom d'affichage et les informations tierces
enable_set_displayname: false
enable_3pid_changes: false
#enable_set_avatar_url: false

# Permet de rechercher des utilisateurs (insensible a la casse)
user_directory:
    enabled: true
    search_all_users: false
    prefer_local_users: true
    show_locked_users: false

# Ajouter des salons (deja existant) a la 1e connexion
# Ne fonctionne que avec des salons publics
# (cela ne fonctionne pas meme si l alias est cree mais que le salon est repasse en prive)
auto_join_rooms:
    - "#Info:exemple.fr"
autocreate_auto_join_rooms: false
autocreate_auto_join_rooms_federated: false
auto_join_rooms_for_guests: false
```

```
# Set variable for OIDC
public_baseurl: https://matrix.exemple.fr

password_config:
   enabled: false
   localdb_enabled: false

# Utiliser notre serveur turn
turn_uris: [ "turns:turn.exemple.fr:443?transport=udp",
"turns:turn.exemple.fr:443?transport=tcp" ]
turn_shared_secret: "..."
turn_user_lifetime: 86400000
turn_allow_guests: true
```

Configuration de nginx

On redirige le port d'écoute par défaut **8008** sur le port **443**. Les clés **ssl** sont renouvelées automatiquement avec **dehydrated** de let's encrypt (installé plus tôt). La création du premier certificat peut se faire avec la commande suivante (qui permet de forcer la génération de certificats avec dehydrated) : `sudo systemctl start dehydrated-renew.service`.

On ajoute donc un fichier de configuration `matrix` dans `/etc/nginx/sites-available` avec la configuration suivante :

```
upstream matrix {
    server localhost:8008;
}

server {
    listen 443 ssl;
    listen [::]:443 ssl;

    # For the federation port
    listen 8448 ssl default_server;
    listen [::]:8448 ssl default_server;

    server_name matrix.exemple.fr;

    include include/ssl_keys_srv-matrix.exemple.fr;

    access_log /var/log/nginx/access.matrix.log;
```

```
    error_log /var/log/nginx/error.matrix.log;

    # Delete "|/_synapse/admin" if Synapse Admin is not use
    location ~ ^(/_matrix|/_synapse/client|/_synapse/admin) {
        # 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://matrix;

        # Include proxy_params file with general parameters
        include include/proxy_params;

        # 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;
    }

    # For the federation configuration
    location /.well-known/matrix/server {
        return 200 '{"m.server": "srv-matrix.exemple.fr:443"}';
        default_type application/json;
    }

    # If Sliding Sync service are installed on our server, let msc3575
    location /.well-known/matrix/client {
        return 200 '{"m.homeserver": {"base_url": "https://srv-matrix.exemple.fr"},
"m.identity_server": {"base_url": "https://vector.im"}, "org.matrix.msc3575.proxy": {"url":
"https://syncv3.matrix.exemple.fr"}}';
        default_type application/json;
        add_header Access-Control-Allow-Origin *;
    }

    # If Element web is installed on our server
    location / {
        return 301 https://element.exemple.fr;
    }

}
```

```
server {
    listen 80;
    listen [::]:80;

    server_name matrix.exemple.fr;
    access_log /var/log/nginx/access.matrix.log;
    error_log /var/log/nginx/error.matrix.log;

    include include/redirect_ssl;
}
```

Synchroniser les dossiers nginx, puis redémarrer les services nginx et synapse :

```
sudo ln -s /etc/nginx/sites-available/matrix /etc/nginx/sites-enabled
sudo systemctl restart nginx
sudo systemctl restart matrix-synapse.service
```

> On peut **vérifier** que le serveur synapse est en **écoute** grâce à l'URL suivante : https://srv-matrix.exemple.fr/_matrix/static/ On peut aussi **vérifier** si notre serveur est **fédéré** grâce à l'URL suivante : https://federationtester.matrix.org/

Installation du client Element

> Pour l'installation, je me suis aidée : → du dépôt GitHub de la version web de Element

On installe Element à partir d'un script nommée `update_element.sh` :

```
sudo apt install jq
sudo mkdir -p /var/www/element
sudo touch /var/www/element/update_element.sh
```

Le script contient les lignes suivantes :

```
#!/bin/sh
set -e

install_location="/var/www/element"
latest="$(curl -s https://api.github.com/repositories/39487546/releases/latest | jq -r
```

```
.tag_name)"

cd "$install_location"

[ ! -d "archive" ] && mkdir -p "archive"
[ -d "archive/element-${latest}" ] && rm -r "archive/element-${latest}"
[ -f "archive/element-${latest}.tar.gz" ] && rm "archive/element-${latest}.tar.gz"

wget "https://github.com/vector-im/element-web/releases/download/${latest}/element-
${latest}.tar.gz" -P "archive"
tar xf "archive/element-${latest}.tar.gz" -C "archive"

[ -L "${install_location}/current" ] && rm "${install_location}/current"
ln -sf "${install_location}/archive/element-${latest}" "${install_location}/current"
ln -sf "${install_location}/config.json" "${install_location}/current/config.json"
```

```
sudo chmod +x /var/www/element/update_element.sh
sudo /var/www/element/update_element.sh
```

Configurer Element à partir du fichier `config.json`, d'abord déplacez-le dans le répertoire `/etc/element` :

```
sudo cp /var/www/element/current/config.sample.json /etc/element/config.json
ln -sf /var/www/element/current/config.sample.json /etc/element/config.json
```

Modifier le fichier `config.json`. Remplacez TOUTE la partie `"default_server_config"` par `"default_server_name"` (ce sont deux méthodes différentes pour la connexion à notre serveur, on utilise la 2e car il va prendre en compte le fichier **.well-known** crée plus tôt dans la configuration nginx de Matrix).

```
"default_server_name": "exemple.fr",
```

Dans le même fichier, on peut personnaliser la page d'accueil et la page home de l'application. Exemple de config possible :

```
"brand": "Element",
"branding": {
    "welcome_background_url":
"https://element.exemple.fr/themes/element/img/backgrounds/lake.jpg",
    "auth_header_logo_url": "https://element.exemple.fr/themes/element/img/logos/logo-
exemple.png",
```

```
    "auth_footer_links": [
        { "text": "Intranet", "url": "https://intranet.exemple.fr/" }
    ]
},
"embedded_pages": {
   "welcome_url": "https://element.exemple.fr/welcome_exemple.html"
},

"default_country_code": "FR",
```

Pour mettre notre propre page d'accueil, on peut copier le fichier par défaut `welcome.html` et modifier certaines choses (comme supprimer le bouton pour s'enregistrer ou modifier la description).

> J'ai rencontré un problème pour le logo, il ne s'affichait pas lorsque je rajoutais la partie `embedded_pages` avec la nouvelle page d'accueil. Donc j'ai directement insérer l'URL du logo dans le fichier `welcome_exemple.html`.

> Sur Element, il y a des features en cours de développement, on peut ajouter la possibilité de les tester (il faut activer la fonctionnalité que l'on veut tester depuis les paramètres), en modifiant une variable dans config.json : `"show_labs_settings": true` Il est aussi possible de forcer l'ajout, voir https://web-docs.element.dev/Element%20Web/config.html#labs-flags

Configuration de nginx

On ajoute un fichier de configuration `element` dans `/etc/nginx/sites-available` avec la configuration suivante :

```
server {
    listen 443 ssl;
    listen [::]:443 ssl;

    server_name element.exemple.fr;

    # TLS configuration, same file for all alias
    include include/ssl_keys_srv-matrix.exemple.fr;

    access_log /var/log/nginx/access.element.log;
    error_log /var/log/nginx/error.element.log;

    # Mise en place de l'icone de l'intranet sur Element
```

```
    location = /vector-icons {
        # Configuration du cache et autres
        include include/static;
        rewrite ^/vector-icons/.*?$
https://element.exemple.fr/themes/element/img/logos/favicon_exemple.ico redirect;
    }

    location / {
        root /var/www/element/current;
        index index.html;
        include include/proxy_params;

        add_header Referrer-Policy "strict-origin" always;
        add_header X-Content-Type-Options "nosniff" always;
        add_header X-Frame-Options "SAMEORIGIN" always;
        add_header X-XSS-Protection "1; mode=block";
    }

}

server {
    listen 80;
    listen [::]:80;

    server_name element.exemple.fr;
    access_log /var/log/nginx/access.element.log;
    error_log /var/log/nginx/error.element.log;

    include include/redirect_ssl;
}
```

Synchroniser les dossiers nginx, puis redémarrer le service nginx :

```
sudo ln -s /etc/nginx/sites-available/element /etc/nginx/sites-enabled
sudo systemctl restart nginx
```

Installation de Synapse Admin

Synapse Admin est une **interface web d'administration** pour le serveur Matrix Synapse. Elle permet notamment de gérer les utilisateurs, les salons ou encore les médias partagés.

>

> Pour l'installation, je me suis aidée : →du README disponible sur le GitHub (steps for 2)

Installer Node.js :

```
curl -sL https://deb.nodesource.com/setup_18.x | sudo bash -
sudo apt install -y nodejs
```

Installer Yarn :

```
curl -sL https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee
/etc/apt/sources.list.d/yarn.list
sudo apt-get update
sudo apt-get install yarn
```

Clôner le Dépôt git et installer les dépendances :

```
sudo useradd -s /bin/bash -m -d /srv/synapse-admin -r synapse-admin
sudo -iu synapse-admin  # Exécuter en tant que synapse-admin
git clone https://github.com/Awesome-Technologies/synapse-admin.git repo
cd synapse-admin
yarn install
yarn start
```

Configuration de nginx

On ajoute un fichier de configuration `synapse-admin` dans `/etc/nginx/sites-available` avec la configuration suivante :

```
upstream synapse-admin {
    server localhost:5173;
}

server {
    listen 443 ssl;
    listen [::]:443 ssl;

    server_name admin.matrix.exemple.fr;
```

```
    include include/ssl_keys_srv-matrix.exemple.fr;

    access_log /var/log/nginx/access.matrix-admin.log;
    error_log /var/log/nginx/error.matrix-admin.log;

    location / {
        proxy_pass http://synapse-admin;

        # Directory where is the repo git of synapse-admin
        root /etc/matrix-synapse/synapse-admin;

        include include/proxy_params;
        client_max_body_size 50M;
    }
}

server {
    listen 80;
    listen [::]:80;

    server_name admin.matrix.exemple.fr;
    access_log /var/log/nginx/access.matrix-admin.log;
    error_log /var/log/nginx/error.matrix-admin.log;

    include include/redirect_ssl;
}
```

Synchroniser les dossiers nginx, puis redémarrer le service nginx :

```
sudo ln -s /etc/nginx/sites-available/synapse-admin /etc/nginx/sites-enabled
sudo systemctl restart nginx
```

Création du service de synapse-admin

Créer un service pour l'automatisation du lancement de l'interface web. Créer le fichier `/etc/systemd/system/synapse-admin.service` :

```
[Unit]
Description=interface web d'administration pour notre serveur synapse

[Service]
```

```
User=synapse-admin
Group=synapse-admi
WorkingDirectory=/srv/synapse-admin/repo

ExecStart=yarn start
RestartSec=10
Restart=always

[Install]
WantedBy=multi-user.target
```

Redémarrer le processus **systemd** pour la prise en compte du nouveau service, puis l'activer au démarrage :

```
sudo systemctl daemon-reload
sudo systemctl enable synapse-admin.service --now
```

Création d'un utilisateur admin

Seuls les utilisateurs administrateurs du domaine peuvent accéder à l'outil Synapse Admin. On va donc créer un utilisateur admin (qui par la suite pourra donner des droits admin à n'importe quel utilisateur).

Créer une clé partagé pour pouvoir créer un nouvel utilisateur (même si l'enregistrement est désactivé) :

```
echo "registration_shared_secret: '$(cat /dev/urandom | tr -cd '[:alnum:]' | fold -w 256 |
head -n 1)'" | sudo tee /etc/matrix-synapse/conf.d/registration_shared_secret.yaml
```

Créer un **nouvel utilisateur administrateur** (il faut rentrer un identifiant, un mot de passe, puis yes) :

```
register_new_matrix_user -c /etc/matrix-synapse/conf.d/registration_shared_secret.yaml
```

Configuration du SSO avec OIDC Connect

Avec le serveur Synapse, on peut directement configurer le SSO pour l'authentification.

> Pour l'installation, je me suis aidée : → de la documentation de OpenId Connect → et plus particulièrement de l'exemple pour LemonLDAP pour la configuration côté Synapse mais

> aussi LemonLDAP

> Forcer d'ajouter les Claims dans le Endpoint Userinfo depuis LemonLDAP (sinon cela ne récupère pas les informations des utilisateurs)

Créer le fichier `/etc/matrix-synapse/conf.d/oidc.yaml` :

```
oidc_providers:
  - idp_id: matrix
    idp_name: "SSO Exemple"
    discover: true
    issuer: "https://sso.exemple.fr/"
    client_id: "matrix.exemple.fr"
    client_secret: "secret_client"
    scopes:
      - "openid"
      - "profile"
      - "email"

    # Uncomment if discover is set to false
    #authorization_endpoint: "https://sso.exemple.fr/oauth2/authorize"
    #token_endpoint: "https://sso.exemple.fr/oauth2/token"
    #userinfo_endpoint: "https://sso.exemple.fr/userinfo"
    #jwks_uri: "https://sso.exemple.fr/.well-known/jwks.json"

    #enable_registration: true
    allow_existing_users: true

    user_mapping_provider:
      config:
        localpart_template: "{{ user.preferred_username }}"
        #confirm_localpart: true
        display_name_template: "{{ user.name }}"
        email_template: "{{ user.email }}"
```

Redémarrer le service synapse :

```
sudo systemctl restart matrix-synapse.service
```

Configuration de MAS (authentication service)

Pour pouvoir se connecter sur Element X quand on utilise le SSO, il faut installer MAS (le successeur de Sliding-Sync).

> Documentation : https://element-hq.github.io/matrix-authentication-service/index.html

> Page en cours d'écriture

Installation de MAS

install_mas.sh

```
ARCH=x86_64
OS=linux
VERSION=latest # or a specific version, like "v0.1.0"

# URL to the right archive
URL="https://github.com/element-hq/matrix-authentication-
service/releases/${VERSION}/download/mas-cli-${ARCH}-${OS}.tar.gz"

# Create a directory and extract the archive in it
mkdir -p /srv/mas
curl -sL "$URL" | tar xzC /srv/mas

# This should display the help message
/srv/mas/mas-cli --help
```

Créer un utilisateur système mas

```
sudo adduser --system mas
sudo addgroup mas
sudo usermod -aG mas mas
sudo usermod -g mas mas
sudo passwd mas
sudo usermod --move-home --home /srv/mas mas
```

```
sudo chown -R mas: /srv/mas/
```

Créer une base de données pour MAS, avec son utilisateur

```
sudo -iu postgres
createuser --pwprompt mas
createdb --owner=mas mas
```

Mettre la configuration de la base de données et la connexion via le SSO dnas le fichier `/etc/matrix-synapse/homeserver.yaml` (car on en aura besoin pour migrer les données en prenant les infos du fichier homeserver.yaml).

Générer un fichier de configuration MAS à partir de notre fichier de conf synapse :

```
sudo /srv/mas/mas-cli config generate --synapse-config /etc/matrix-synapse/homeserver.yaml --
output /srv/mas/mas_config.yaml
```

Ajuster le fichier mas_config.yaml pour qu'il ressemble à ceci :

```
# En cours de rédaction
```

- Modifier le fichier nginx .well-known
- Ajouter un fichier nginx pour MAS
- Éteindre les serveurs synapse et MAS Commandes pour tester les configurations des deux cotes avant la migration

```
sudo /srv/mas/mas-cli syn2mas check --config /srv/mas/mas_config.yaml --synapse-config
/etc/matrix-synapse/homeserver.yaml
sudo /srv/mas/mas-cli syn2mas migrate --config /srv/mas/mas_config.yaml --synapse-config
/etc/matrix-synapse/homeserver.yaml --dry-run
```

- homeserver.yaml : ajout de la partie msc3861
- migration :

```
sudo /srv/mas/mas-cli syn2mas migrate --config /srv/mas/mas_config.yaml --synapse-config
/etc/matrix-synapse/homeserver.yaml
```

- changer la configuration de notre SSO pour les adresses autorisées
- supprimer la partie oidc_provider dans homeserver.yaml
- lancer les services matrix-synapse et MAS
- vérifier que tout est bon avec la commande suivante :

```
sudo /srv/mas/mas-cli doctor --config /srv/mas/mas_config.yaml
```

> Et la cela réussi à avoir des infos sur mon compte, mais je n'arrive pas à me connecter {.is-warning}

Mise à jour

Serveur Matrix Synapse

> Source : https://element-hq.github.io/synapse/latest/upgrade.html

```
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
sudo systemctl restart matrix-synapse.service
```

Vérifier que la mise à jour a fonctionné :

```
curl http://localhost:8008/_synapse/admin/v1/server_version
# {"server_version":"1.121.1"}
```

Client Element

Lancer le script /var/www/element/update_element.sh

> Le Script ne déplace pas les fichiers de customisation (logos de ville, page d'accueil...). En l'attente de l'automatisation par le script. Il faut copier les fichiers : → welcome_exemple.html → themes/element/img/logos/* → themes/element/img/background/*

Synapse admin

> Source : https://github.com/Awesome-Technologies/synapse-admin?tab=readme-ov-file#steps-for-2

```
sudo systemctl stop synapse-admin
sudo -iu synapse-admin
git pull
yarn install
yarn start
exit
sudo systemctl start synapse-admin
```

> Autrice originale de cette documentation : Annabelle Cortès

---

Révision #7
Créé 22 juin 2025 08:44:15
Mis à jour 22 juin 2025 09:21:45