Fleury Lucie
Rousseau Matthieu
Millies-lacroix Aurelien
Landier Clément
Mouflier Guillaume

Déploiement d'un nœud Matrix pour la FIPA

Table des matières

I – Présentation du projet 4
II – Présentation de MATRIX : 4
A – Matrix : 4
B – Histoire : 4
C – Présentation de docker : 5
D – Ansible : 5
E – Fédérations : 6
F – Bridges : 7
III – Analyse fonctionnelle 8
Analyse du besoin : 8
Analyse Fonctionnelle du Besoin : 8
Analyse Fonctionnelle Technique : 9
IV – Serveurs 11
V – Mise en place de Matrix 12
A – Choix de l'OS : 12
B – Choix de l'implémentation 12
C – Installation de Matrix sur les différents OS : 13
1 – Ubuntu : 13
2 – Rocky Linux : 14
3 – Debian : 15
VI – Docker/Ansible : 16
A - Docker sur Raspberry 16
B - Docker sur Debian : 17
Migration d'une base de données vers matrix-docker-ansible-deploy 17
VII – Mise en place des bridges : 18
Puppet : 18
Configuration du Bridge Facebook / Messenger : 19
Utilisation du bridge facebook : 20
Configuration du Bridge Discord : 22
VIII – Conclusion : 23
IX – Sources : 24

I – Présentation du projet

L'objectif du projet est la mise à disposition d'un nœud Matrix dédié à la FIPA afin de faciliter les échanges internes à la formation dans un cadre sécurisé et intuitif. Au-delà de l'aspect purement fonctionnel, il s'agit d'une opportunité de présenter le monde de l'open source à la FIPA.

Dans nos efforts pour tendre vers cet objectif, nous avons dû :

- Nous approprier le travail réalisé l'année précédente,
- Déployer une solution d'hébergement pérenne,
- Réaliser des passerelles vers des applications moins sécurisées mais plus populaires afin de faciliter la transition.

II – Présentation de MATRIX :

A – Matrix :

Matrix est un projet open source, combinant un protocole et de nombreuses implémentations, permettant la communication en temps réel. Il s'agit d'un protocole libre de droit et léger. Il a été conçu pour permettre la communication en ligne, la voix sur IP ainsi que la visioconférence aux utilisateurs sans passer par un service tiers.

Matrix s'adresse aux utilisateurs recherchant la sécurité et la garantie de leur vie privée. S'agissant d'un format ouvert (open standard), n'importe quel serveur respectant les spécifications du protocole Matrix peut communiquer avec n'importe quel autre, et l'utilisation de ces serveurs peut se faire par l'intermédiaire de n'importe quel client. Cette flexibilité impressionnante est une marque de fabrique de l'open source ; le protocole étant ouvert, tout un chacun peut en écrire une implémentation.

Il est également possible d'interfacer le serveur Matrix avec d'autres logiciels, dont certains permettent de relier Matrix à d'autres messageries instantanées. Il s'agit de passerelles.

B – Histoire :

Le projet Matrix a été créé en 2014 par Amdocs. Il s'agit d'une entreprise multinationale spécialisée dans les services et logiciels pour différents fournisseurs de services. Le projet avait pour nom « Amdocs Unified Communications ».

En 2015, voyant le succès du protocole, l'entreprise décide de mettre en place une filiale « Vector Creations Limited » dédiée au seul développement du projet Matrix. L'entreprise Amdocs finance alors entièrement le développement du protocole, qui s'est achevé en 2017.

En 2017, Amdocs stoppe le financement du projet et l'équipe de Vector Creations Limited se sépare de sa maison mère, créant sa propre organisation pour continuer le développement de Matrix.

En 2020, le client développé par l'équipe (Riot) est renommé Element. L'entreprise est renommée de la même manière. Dans la même année, Element remporte le plus gros contrat au monde pour un service logiciel collaboratif.

C – Présentation de docker :

Docker est une technologie de conteneurisation qui permet d'empaqueter une application avec tous ses dépendances pour l'exécuter sur tout type de serveur. Docker permet donc de faire fonctionner une application sans avoir à se soucier de l'environnement : un conteneur Docker est un microsystème isolé avec son propre noyau, ses propres fichiers et ses propres processus.

Nous avons utilisé Docker afin de pouvoir facilement déployer notre projet sur un serveur Debian. Nous avons ainsi pu aisément gérer l'installation de nos différentes applications et faciliter la mise en place du serveur.

D – Ansible :

Ansible est un outil d'automatisation open source qui peut être utilisé pour configurer et gérer les ordinateurs et les serveurs. Ansible peut être utilisé pour déployer des applications et des services sur des serveurs (ce qui va nous servir ici), et pour gérer les tâches de configuration et de maintenance.

Ansible utilise un langage simple qui permet aux utilisateurs de définir les tâches à automatiser. Les tâches sont décrites dans des fichiers de configuration, nommés playbooks, en YAML.

E – Fédérations :

Une fédération est un groupe de fournisseurs de services informatiques ou de réseaux qui conviennent de normes de fonctionnement de manière collective.

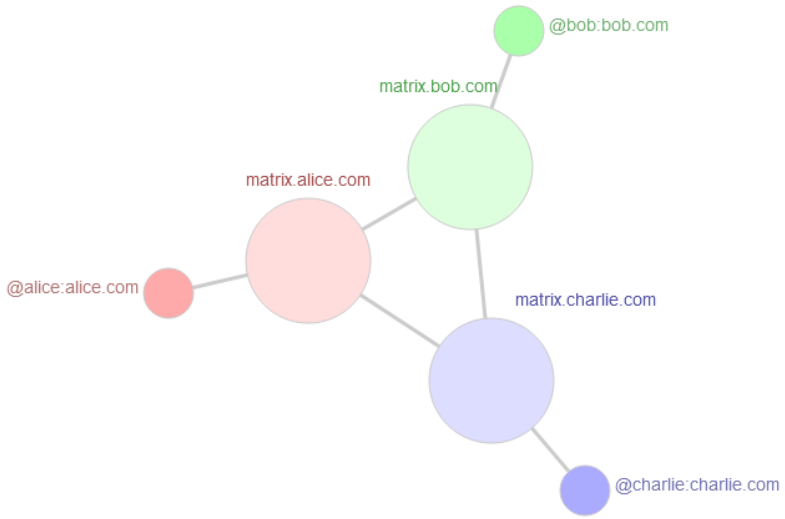
Le terme peut être utilisé pour décrire l'interopérabilité de deux réseaux de télécommunications distincts, formellement déconnectés, qui peuvent avoir des structures internes différentes. Le terme "nuage fédéré" fait référence à la facilitation de l'interconnexion de deux ou plusieurs nuages informatiques géographiquement séparés.

Dans les environnements réseau, être fédéré signifie que les utilisateurs peuvent envoyer des messages d'un réseau à l'autre. C'est différent du fait d'avoir un client qui peut fonctionner avec les deux réseaux, mais qui interagit avec les deux indépendamment.

Deux exemples de protocole qui permet la création de fédérations sont les protocole SMTP et XMPP. En effet, ils permettent des utilisateurs de services différents (ex : Gmail et Outlook) de communiquer entre eux sans que les systèmes originaux ne soit les mêmes.

Matrix peut être configuré pour la fédération ou non. Cela permet d'ouvrir les salons à des utilisateurs d'un autre serveur, ou au contraire de rester isolé. Ainsi même deux serveurs basés sur des implémentations différentes sont capables de communiquer. Cela est rendu possible par le strict respect des spécifications du protocole Matrix, qui prévoit entrées et sorties d'informations universelles. Ainsi la structure fédérée fonctionne à la manière d'une API : les données doivent être agencées de la même manière en entrée en sortie d'un nœud, mais peu importe la manière dont elles sont traitées dans le nœud. Cela permet à un serveur d'obtenir les informations (messages, utilisateurs...) de sorte qu'il puisse les utiliser.

Du point de vue de l'utilisateur on ne se connecte qu'à un serveur, mais en réalité ce serveur est lui-même lié aux autres serveurs de la fédération. Dans l'exemple ci-dessous, bob peut ainsi parler avec Alice même si les deux sont connectés à deux serveurs distincts – et même si les deux serveurs utilisent des implémentations différentes du protocole Matrix.



F – Bridges :

Comme nous avons pu le voir matrix est un service de chat extensible qui permet à des utilisateurs de services différents de communiquer entre eux même si les implémentations du protocole peuvent diverger.

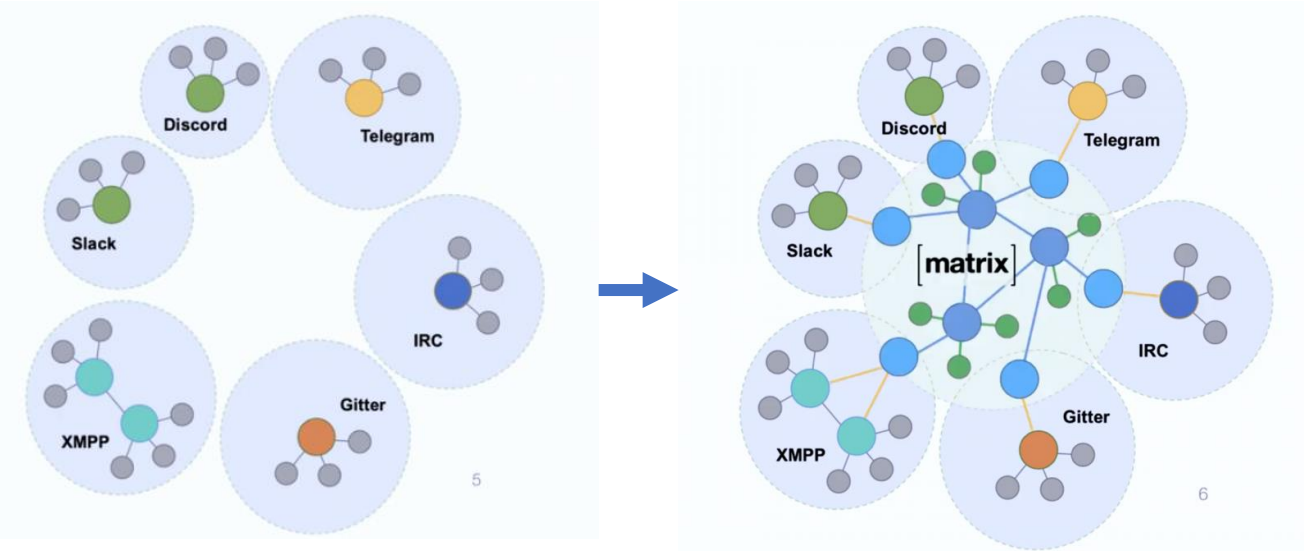
Cependant il peut arriver que nous voulions communiquer avec des applications de chat n'utilisant pas le protocole Matrix. Pour résoudre ce problème il existe une solution : les bridges

Un bridge Matrix est un moyen de connecter un service de chat existant (comme Slack) à Matrix, afin que les utilisateurs du service de chat puissent communiquer entre eux, quel que soit le service qu'ils utilisent.
Pour ce faire, un bot se trouve à la fois dans le chat de Matrix et dans l'autre service de chat, et relaie les messages entre les deux.

Cela permet aux utilisateurs de différents services de discussion de communiquer entre eux, sans avoir à changer de service.

Les passerelles sont utiles pour de nombreuses raisons. Par exemple, elles peuvent être utilisées pour connecter des personnes qui utilisent des services de chat différents, permettant de centraliser les messageries, ou pour connecter des personnes qui utilisent le même service de chat mais se trouvent dans des pays différents.

Dans notre cas, les passerelles nous permettent de communiquer avec les utilisateurs d'outils de communication déjà utilisés tels que Discord ou Messenger, cela permet de faire un lien entre la communication sur le groupe Messenger et Discord afin de ne pas obliger les utilisateurs à changer leurs habitudes et sur le plus long terme de faciliter l'adoption d'une connexion directe au serveur Matrix.



III – Analyse fonctionnelle

Analyse du besoin :

Dans un monde connecté, nous avons plusieurs outils permettant de communiquer à longue distance. Matrix est un outil qui nous permet de réaliser cela. Actuellement, la communication entre les promotions des FIPA est très sommaire. Cet outil nous permettra d'interconnecter les différentes promotions de la FIPA et de permettre aux nouveaux étudiants une meilleure intégration dans l'école.

Cahier des charges :

Matrix est une application open source permettant de mettre en place une communication sécurisée. Pour cela, Matrix nous propose un système de messagerie encrypté point à point avec des conversations synchronisées. Cette application intègre aussi un système VoIP permettant d'effectuer des appels sécurisés. Matrix dispose d'un système de « Bridging » qui va nous permettre de lier l'application à d'autres messageries en temps réel telles que Telegram, Signal ou encore Discord. Le but dans ce projet est de mettre en place un serveur Matrix capable d'assurer des communications textuelles et vocales, ainsi que de lier l'application à d'autres messageries.

Permettre à l'utilisateur d'utiliser le système de messagerie sur ordinateur et téléphone (si possible).

Analyse Fonctionnelle du Besoin :

Notre serveur va permettre d'échanger des messages entre deux personnes et ceux de manière sécurisée. De plus ce service nous permettra de joindre d'autres services tels que Discord et Telegram.

Notre service peut être utilisé sur un téléphone mobile comme sur un ordinateur ou tout autre appareil possédant un navigateur Web. L'idée est d'avoir ici un outil facile d'utilisation et accessible par tous les étudiants de la FIPA.

Phase de vie du produit :

- Mise en place du service : Mise en place de Matrix et de ses intégrations (passerelles vers Telegram, Signal, WhatsApp, email et Discord).
- Montage et déploiement du service : Déploiement sur machine virtuel et création des comptes utilisateurs
- Utilisations du produit par les élèves de la FIPA.
- Maintenance et mise à jour : En cas de panne ou d'erreur, intervention et amélioration pour l'avenir.
- Destructions des comptes utilisateurs qui n'existent plus.

Éléments du milieu extérieur : Location d'un serveur et d'un nom de domaine.

Analyse Fonctionnelle Technique :

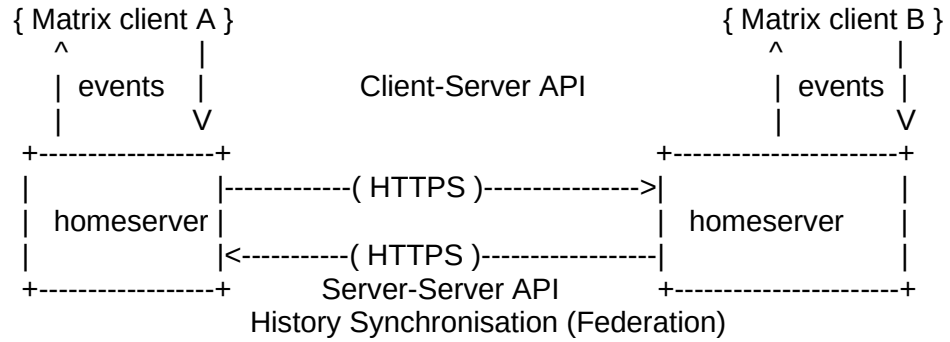
Comment fonctionne Matrix :

Matrix définit des API pour synchroniser des objets JSON extensibles appelés "événements" entre des clients, des serveurs et des services compatibles. Les clients sont généralement des applications de messagerie/VoIP ou des dispositifs/hubs IoT et communiquent en synchronisant l'historique des communications avec leur "serveur domestique" à l'aide de l'API client-serveur". Chaque serveur domestique stocke l'historique de communication et les informations de compte de tous ses clients, et partage les données avec l'écosystème Matrix plus large en synchronisant l'historique de communication avec d'autres serveurs domestiques et leurs clients.

Les clients communiquent généralement entre eux en émettant des événements dans le contexte d'une "salle" virtuelle. Les données de la salle sont répliquées sur tous les serveurs domestiques dont les utilisateurs participent à une salle donnée. Ainsi, aucun serveur domestique n'a le contrôle ou la propriété d'une salle donnée. Les serveurs domestiques modélisent l'historique des communications sous la forme d'un graphe partiellement ordonné d'événements, appelé "graphe d'événements" de la salle, qui est synchronisé avec une cohérence éventuelle entre les serveurs participants à l'aide de l'"API serveur-serveur". Ce processus de synchronisation de l'historique des conversations partagées entre des serveurs domestiques gérés par différentes parties est appelé "Fédération". Matrix optimise les propriétés de disponibilité et de partitionnement du théorème CAP au détriment de la cohérence.

Par exemple, pour que le client A envoie un message au client B, le client A exécuté un HTTP PUT de l'événement JSON requis sur son serveur domestique (SD) en utilisant l'API client-serveur. Le SD de A ajoute cet événement à sa copie du graphe d'événements de la pièce, en signant le message dans le contexte du graphe pour en assurer l'intégrité. Le SD de A réplique ensuite le message au SD de B en effectuant un HTTP PUT à l'aide de l'API serveur-serveur. Le SD de B authentifie la requête, valide la signature de l'événement, autorise le contenu de l'événement puis l'ajoute à sa copie du graphe d'événements de la salle. Le client B reçoit ensuite le message de son serveur domestique via une requête GET à longue durée de vie.

Comment les données circulent entre les clients :



Chaque client est associé à un compte utilisateur, qui est identifié dans Matrix par un "ID utilisateur" unique. Cet ID est associé au serveur domestique qui a attribué le compte et a la forme suivante : @IDuser:nomdedomaine

La spécification Matrix donne un sens particulier au terme "dispositif". En tant qu'utilisateur, je peux avoir plusieurs dispositifs : un client de bureau, des navigateurs web, un dispositif Android, un iPhone, etc. Ces termes se rapportent en gros à un dispositif réel dans le monde physique, mais vous pouvez avoir plusieurs navigateurs sur un dispositif physique, ou plusieurs applications client Matrix sur un dispositif mobile, chacun d'entre eux étant son propre dispositif.

Les dispositifs sont principalement utilisés pour gérer les clés utilisées pour le cryptage de bout en bout (chaque dispositif obtient sa propre copie des clés de décryptage), mais ils aident également les utilisateurs à gérer leur accès - par exemple, en révoquant l'accès à des dispositifs particuliers.

Lorsqu'un utilisateur utilise un client pour la première fois, celui-ci s'enregistre comme un nouveau dispositif. La longévité des dispositifs peut dépendre du type de client. Un client web abandonnera probablement tout son état lors de la déconnexion et créera un nouveau dispositif à chaque fois que vous vous connecterez, afin de garantir que les clés cryptographiques ne seront pas transmises à un nouvel utilisateur. Dans un client mobile, il peut être acceptable de réutiliser le dispositif si une session de connexion expire, à condition que l'utilisateur soit le même.

Les dispositifs sont identifiés par un device_id, qui est unique dans le cadre d'un utilisateur donné.

Un utilisateur peut attribuer un nom d'affichage lisible par l'homme à un dispositif, pour l'aider à gérer ses dispositifs.

IV – Serveurs

Afin d'installer Matrix nous avons besoin d'un serveur :

Voici la configuration minimale pour réaliser le projet :

- 4 vCPU
- 8 GB RAM
- 400GB de stockage.

Cependant nous avons rencontré quelques contraintes pour mettre celui-ci en place.

En effet, celui-ci ne pouvait être financer par l'école et dans le cadre ou le service seras maintenue durant des années pour les futures générations d'étudiant FIPA. Il nous était impossible de l'installer sur une VM, ou un autre produit nous appartenant (Rapberry PI). Bien sûr dans le cadre des tests et de la mise en place du service nous avons utilisé nos propres appareils. Notre première idée a donc été de proposer un financement partagé, payer par tous les étudiants de la FIPA.

Nous avions trouvé un serveur répondant à nos besoins proposés sur le site "Contabo" au prix de 93.46 € par an.

Il nous fallait aussi un nom de domaine proposé au prix de 6 € par an.

Comme nous sommes une soixantaine d'élèves en FIPA le coût total par personne aurais état de 1.70 € par an. Cela n'a malheureusement pas pu être mis en place. En effet avec tous les services de communication actuel : Discord, Telegram, Whatsapp, et bien d'autres. Il est difficile de convaincre des étudiants d'investir dans un service de communications qu'il ne connaisse pas, et qu'ils n'ont pas l'habitude d'utilisé. L'idée d'un financement participatif a donc était abandonné.

Dans un second temps, nous avons demandé aux associations de réseau tel que Minet d'héberger notre service sur leur serveur. L'association possédant déjà plusieurs services et une place limité a alors refuser. Nous avons réitéré cette expérience avec les autres associations possédant un serveur.

Nous avons finalement trouvé un serveur dans une de nos entreprises, la société se débarrassais de ce dernier. Nous l'avons donc récupéré pour Matrix. Ce dernier possédait la configuration nécessaire au déploiement de Matrix. La dernière contrainte se présentant à nous était l'endroit où placer le serveur. Nous avons donc demandé aux associations s'il était possible d'installer notre serveur dans leur rac, ce qui a était de nouveau refusé. Ne trouvant aucun endroit ou le placer nous avons opter pour une autre solution.

Nous avons demandé à la DISI s'il était possible de déployer une VM pour notre serveur. Celle-ci a accepté notre demande et nous a fourni une machine pour le déploiement de Matrix.

V – Mise en place de Matrix

A – Choix de l'OS :

Pour installer notre serveur nous avons le choix entre différent OS.

Ubuntu : C'est l'OS classique qui nous permet d'installer Matrix et de faire les configurations minimales. Nous l'avons utilisé dans un premier temps pour prendre en main Matrix

EL8 : Nous avons essayé d'installer Matrix sous Rocky Linux 8, il a été très difficile de le mettre en place et l'effort n'en valais pas la récompense. Nous avons donc abandonné cette distribution.

Debian : Sous Debian nous pouvons utiliser docker. Il existe un dossier docker pour Matrix qui nous permet d'auto-installer tous les services en lien avec Matrix, attention les configurations ne se ferons pas automatiquement pour tous les services. C'est l'option pour laquelle nous avons optée une fois la prise en main de Matrix terminé

B – Choix de l'implémentation

Il existe plusieurs implémentations du protocole matrix, celles que non avons considérer sont :

- Synapse, il s'agit de l'implémentation original de matrix développer par l'équipe Matrix en python3 et qui reste à ce jour l'implémentation la plus utiliser.

- Dendrite, il s'agit d'une implémentation en Go développer par l'équipe Matrix qui a pour but d'être plus efficace, stable, évolutif et qui se décrit comme un serveur de seconde génération, cependant le projet est en beta et ne tient pas encore c'est promesse.

- Ligasen un fork de Dendrite apparemment utiliser par institutions financière en chine, cependant l'implémentation n'est que très peut utiliser hors de ce contexte et ne permet que peu de modularité.

- Construct et Conduit, qui sont deux implémentations, respectivement en C++ et Rust qui se concentre sur les performances, cependant elles n'apportent pas d'autre bénéfices et sont toujour en beta.

Nous avons choisi d'utiliser l'implémentation par default car il s'agit de la plus complète, et évolutive de plus elle est utilisée par la majorité de la communauté Matrix, et sera donc plus facile à maintenir.

C – Installation de Matrix sur les différents OS :

1 – Ubuntu :

La distribution Ubuntu est la distribution la mieux documenté pour mettre en place Matrix. Nous avons donc tous essayer avec cette distribution pour prendre en main l'installation de Matrix. Nous n'avons pas rencontré de gros problèmes même si certaine difficulté ce son présenté.

Pour déployer Matrix sous Ubuntu, nous avons déployé une machine virtuelle possédant la configuration minimale a sa mise en place. Cette dernière ne possédait pas le stockage adéquat, comme nous l'utilisons pour effectuer des tests nous n'avons pas besoin d'un gros stockage.

Nous avons ensuite suivi la documentation de Maxime Monteforte, élèves qui avais initialement travailler sur la mise en place de Matrix.

Afin d'installer Matrix sous Ubuntu nous avons installé tous les paquets nécessaires au bon fonctionnement de Matrix.

Puis nous avons éditer les fichiers de configurations requis, nous avons rencontré aucun problème à ce niveau-là.

Matrix a était installé sans soucis et nous avons mis en place une base de données avec mysql, base de données permettant de stocker tous les comptes utilisateurs possible.

2 – Rocky Linux :

Rocky Linux a été l'OS sur laquelle nous sommes partie à l'aventure, c'est un OS que certains d'entre nous connaisse bien car il l'utilise dans leur quotidien. Rocky Linux a l'avantage d'être basé sur CentOS, c'est une distribution qui est donc plus destiné au serveur et nous pensions que ce serais une bonne idée de le déployer dessus.

La documentation permettant de mettre en place Matrix sur Rocky Linux est faible, ne peut de personne on essayer et c'est donc mal documenté, difficile à mettre en place et difficile par conséquent à maintenir.

Nous avons rencontré notre premier problème lorsque nous avons cherché à installer les paquets prérequis a l'installation de Matrix. Les paquets nécessaires se trouve dans des repo qui sont désactiver par défaut sur CentOS. Ils nous ont donc fallu chercher et activer tous les repos dont nous avions besoin pour installer Matrix.

Nous avons par la suite modifié les options du pare-feu pour autoriser les ports utilisés par Matrix

Quelques soucis on était rencontré pendant les prérequis, notamment avec les repos qui n'étais pas activé. Par la suite nous avons mis en place Matrix.

Pour l'installation de Matrix, nous avons créé un environnement virtuel. Un environnement virtuel est un répertoire qui vas contenir l'installation de Matrix. Nous n'avions encore jamais travaillé sur un environnement semblable. Nous avions donc mal compris l'utilité de cet outil et tous ces fonctionnements. L'installation s'est déroulée sur plusieurs séance, notre machine a donc était éteinte et nous ne savions pas comment accéder à l'environnement virtuel que l'on avait créé.

Durant l'installation des paquets Matrix, la documentions suivi n'étais pas à jour, nous avons donc installer des paquets non utilisés que nous avons dû supprimer de notre machine pour que tout soit propre. Nous avons pu trouver par la suite comment installer le paquet Matrix sur CentOS8 sur le github officiel de Matrix.

Une fois notre environnement retrouvé, nous avons pu reprendre notre configuration, lors de celle-ci peut de problème on était rencontré. Seuls les certificats nous en poser des problèmes. Dans la documentation, des certificats personnels était utilisé. Ce qui n'est pas notre cas, nous avons dû trouver le chemin des certificats par défaut de Matrix pour le faire fonctionner.

Une fois la configuration faites nous n'avions plus qu'à tester le serveur Matrix sur une page Web, cependant une erreur dans la configuration a était faites ce qui nous a poser plusieurs problèmes pour tester. Une fois l'erreur corriger, cela ne fonctionner toujours pas. Après quelques recherches nous avons dû modifier notre fichier /etc/hosts pour pouvoir accéder à notre serveur.

Des recherches ont commencé à être effectuer pour déployer le reste des fonctionnalités de Matrix, aux vues du temps passer et des efforts fournis par rapport aux camarades travaillant sur une machine Ubuntu nous avons décidé d'arrêter d'utiliser cette distribution.

Pour conclure sur la machine Rocky Linux, après avoir constaté que nous rencontrions plus de problème sur celle-ci que pour les autres distributions. De plus celle-ci demande plus d'effort de compréhension, plus de travail de maintien et plus de recherche au vu de la faible documentation présente. Nous avons donc décider d'abandonner cette distribution. Les personnes travaillant dessus on par la suite installer Matrix sur une machine Ubuntu pour posséder les mêmes compétences que les autres.

3 – Debian :

Il nous est aussi possible d'installer Matrix sous Debian, la distribution étant porche d'Ubuntu, la procédure d'installation ne change pas trop. Nous n'avons donc pas rencontré de problèmes. Cependant c'est cette distribution qui a était retenue pour mettre en place notre serveur final car celle-ci est plus stable qu'Ubuntu et convient donc mieux à un serveur de production.

Toutes les documentations ainsi que les procédures d'installation sont présentes sur notre Gitlab, il est donc possible de reproduire l'ensembles de notre travail. Attention cependant, la documentation pour Rocky Linux s'arrête là où nos recherches se sont arrêté.

VI – Docker/Ansible :

Pour faciliter la mise en place d'un serveur matrix ainsi que sa maintenance, un playbook ansible a été créée afin d'automatiser l'installation et la configuration du serveur synapse ainsi que tous les services pouvant être ajouter au serveur. Ce playbook utilise des conteneurs docker, ce qui permet de déployer Matrix sur n'importe quel serveur indépendamment de l'environnement. Mais également d'ajouter un service simplement en ajoutant un conteneur. Tout cela se fait via un fichier de configuration vars.yml situé dans un répertoire portant le nom du serveur (matrix.domaine). Cela permet de déployer plusieurs serveur synapse sur une même machine (chose que ne sera pas effectué dans ce projet).

A - Docker sur Raspberry

Une possibilité du déploiement de Matrix était de faire tourner le service sur une raspberry. Les raspberry étant de petits ordinateurs permettant de mettre en place des serveurs pour des projets. Ces machines utilisent leur propre distribution "raspbian" basé sur Debian, OS que l'on utilise pour faire tourner Docker.

Afin de tester une mise en place de matrix-synapse via matrix-docker-ansible-deploy et écrire la documentation, un serveur matrix a été déployé sur un Raspberry PI 4 sur Raspberry OS 64bits. Le deploy ayant initialement été prévu pour une architecture amd64, effectuer ce test sur Raspberry nous a donc mis face à certain problème qui n'ont pas été rencontré lors du véritable déploiement.

Dans un premier temps, comme il ne s'agissait pas d'une nouvelle installation propre, docker était déjà installé sur la machine, or cela a créé un conflit de sources avec la source mise en place par ansible. Pour régler ce problème, une installation propre de l'OS a été faite. Cependant, cela peut également être résolu en supprimant le fichier /etc/apt/sources.list.d/docker.list.

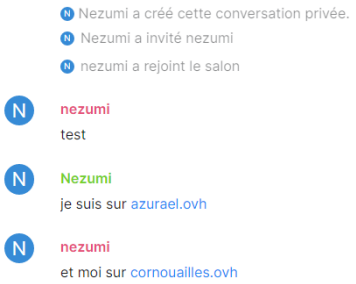
Matrix-docker-ansible-deploy fonctionne via des images préconstruites majoritairement pour amd64. Certaines images ont été construites pour d'autres architecture mais pas toute, ce qui va obliger ansible à construire les images manquantes. Ce qui va nous poser des problèmes car certaines images ne vont pas réussir à se construire correctement, comme element et matrix-registration.

Pour matrix-registration, une erreur apparaissait au démarrage indiquant que le service n'a pas réussi à démarrer. Après vérification dans les logs, il semblerait qu'il s'agisse d'un problème de dépendance python. Nous avons donc essayé de construire manuellement l'image avec la dépendance mais chaque essaie s'est soldé par un échec. Comme il ne s'agit que d'un test et que le problème n'a pas été rencontré sur le déploiement final, nous n'avons pas trouvé de solution à ce problème.

En ce qui concerne element, la construction de l'image fonctionne correctement mais le client web n'affiche d'une page blanche. Aucun message d'erreur permettant de trouver l'origine du problème n'a été trouvé. Nous avons donc simplement utilisé un client différent (cinny) qui n'a rencontré aucun problème.

L'architecture est spécifiée dans le fichier de configuration avec la ligne : matrix_architecture: <your-matrix-server-architecture>

La mise en place de ce serveur en plus du serveur FIPA nous a permis de vérifier le bon fonctionnement des fédérations. On peut voir ici une conversation entre 2 comptes sur les 2 serveurs différents



B - Docker sur Debian :

Notre installation a finalement été effectué sur un serveur debian en architecture amd64 et nous a donc permis d'éviter les problèmes mentionnés précédemment. Cependant, cela ne veut pas dire que nous n'en avons pas rencontré.

Migration d'une base de données vers matrix-docker-ansible-deploy

Afin de documenter une migration d'une base de données existante vers un nouveau serveur matrix déployé à l'aide d'ansible, J'ai donc essayé de migrer la base de données Postgres précédemment utilisé lors d'une autre installation de synapse, qui a elle-même été migré depuis SQLite.

Il a donc fallu faire un dump de la base de données existante pour l'importer sur le nouveau serveur à l'aide du playbook ansible. Cependant, le nom du propriétaire de la base n'était pas le même et a donc dû être changé dans le fichier dump à l'aide de sed.

Mais après cela, une nouvelle erreur est apparue concernant une relation déjà existante.

```
stderr: 'ERROR:  relation "access_tokens" already exists'
```

Il s'agissait apparemment d'une erreur connue et il fallait supprimer le contenu du répertoire postgres et réinitialiser le service. Mais cela n'a pas réglé le problème. Après quelques recherches, je n'ai pas trouvé de solutions pour résoudre cette erreur.

Je me suis donc connecté à la base de données via postgres et j'ai en effet remarqué l'existence de la relation indiquée dans le message d'erreur. Celle-ci étant vide, j'ai supprimé la table pour vérifier si cela résoudrait le problème. Pour éviter tout problème j'ai également effectué la migration directement dans le docker et non via le playbook (chose qui ne marchait pas non plus avant). Et la migration s'est effectué sans erreur. Pour être sûr d'effectué la migration proprement, j'ai à nouveau supprimé les données du répertoire postgres, réinitialisé le service postgres puis lancé le playbook pour effectuer la migration. Celle-ci s'est effectué sans problèmes.

VII – Mise en place des bridges :

Une fois notre serveur Matrix mis en place, nous avons cherché à déployer des bridges. Des bridges sont des passerelles vers d'autres application de communication, comme Messenger, Discord ou Telegram. Le but des bridges est de pouvoir communiquer de notre client Matrix vers des applications externe et cela de manière chiffrée.

Dans notre projet, nous nous sommes concentrés sur la mise en place de bridge vers l'application Messenger ainsi que l'application Discord.

Nous avions plusieurs méthodes pour mettre cela en place, soit avec une configuration manuelle, avec une configuration automatique des fichiers de configuration ou avec un outil de centralisation de configuration Puppet.

Puppet :

Durant nos recherches pour établir les bridges, nous avons regardé pour utiliser Puppet. Puppet est un outil de centralisations de configuration. C'est un outil qui va permettre de propager un fichier de configuration sur notre serveur. Plusieurs documentations sur les bridges utilisent ce système. Pour mettre en place Puppet nous devons respecter l'architecture suivante :

GitLab (Fichier de configuration) --> Serveur Puppet --> Serveur Matrix

Cela nous oblige donc à créer un répertoire sur notre GitLab dans lequel nous mettons tous les fichiers de configurations. Cela est faisable, mais le répertoire devra être transporter, il ne peut pas rester sur le GitLab personnel d'un membre du groupe. Le serveur Puppet iras chercher sur le GitLab les configurations et les propageras sur le serveur Matrix. Cela implique donc que nous devons déployer un deuxième serveur dédié à Puppet. Comme nous avons rencontré quelques problèmes pour déployer un serveur, cela nous semble difficile d'en déployer un deuxième. De plus Puppet est un outil que l'on utilise surtout dans une grande infrastructure, cela nous permet de réinstaller des machines sans trop réfléchir car tous les fichiers de configurations sont déjà réalisés.

Nous n'allons donc pas utiliser toutes les fonctions de Puppet mais uniquement le double puppeting. Le double puppeting gère les configurations des utilisateurs de chaque côté du bridge, ce qui permet de remplacer le compte par défaut du bridge par le compte matrix de l'utilisateur. Par conséquent, lorsqu'un message est envoyé depuis une autre messagerie, tel que messenger ou telegram, il sera envoyé depuis votre compte matrix et non l'utilisateur par défaut du bridge, dans notre cas les bots.

Configuration du Bridge Facebook / Messenger :

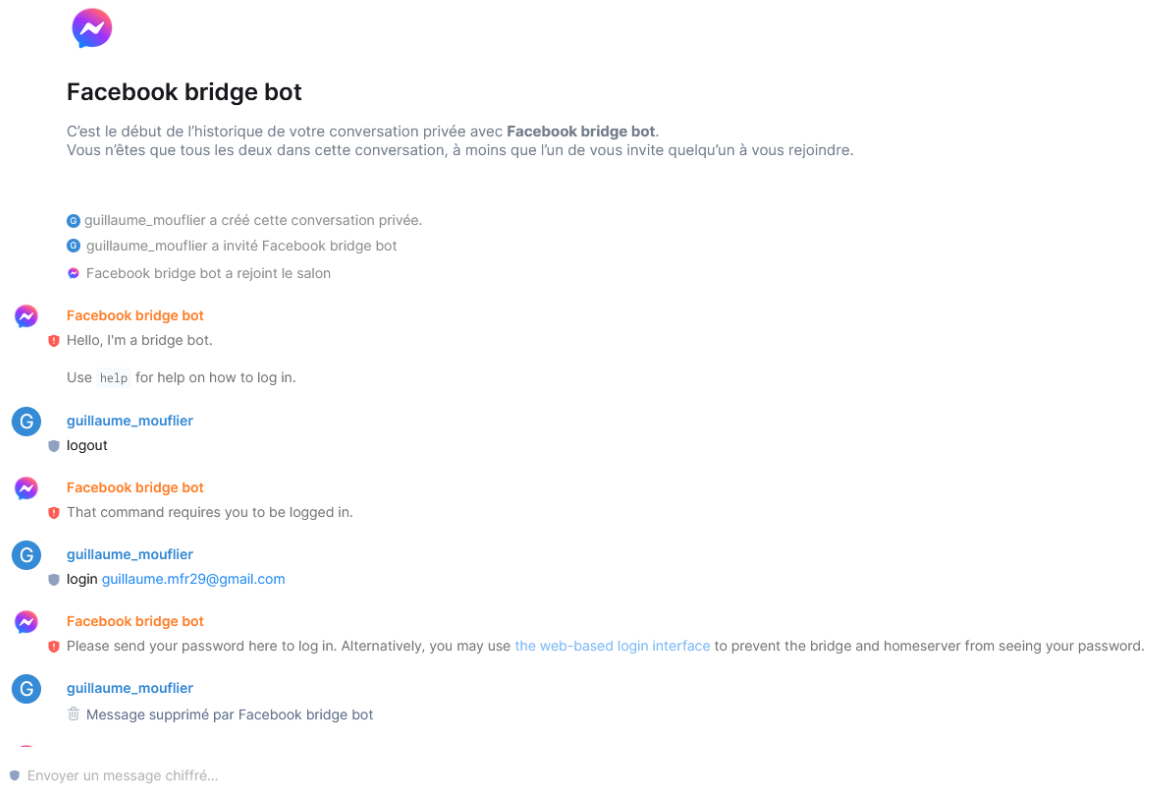
Le bridge Messenger est le premier bridge que nous avons essayé de mettre en place. Dans un premier temps, nous avons essayé d'installer le bridge en éditant nous même les fichiers de configuration sur notre serveur. Pour cela, nous avons suivi une documentation GitHub détaillant le contenue des fichiers de configuration. Après avoir modifié le fichier, une erreur de communication avec notre base de données apparaissait. Celle-ci était injoignable. Nous avons donc approfondi nos recherches et nous avons remarqué que la syntaxe requis dans la documentation n'était pas bonne, et qu'a la place de notre nom de domaine nous devions mettre notre adresse IP. Une fois ceci modifié nous avons réessayer de set up le bridge pour Messenger. Une erreur de communications avec la base de données restait présente. Nous avons donc abandonné l'idée de mettre en place le bridge manuellement car il y avait trop de paramètres qu'on ne savait pas modifier et qui étais plus ou moins documenté. Nous avons donc choisi de mettre en place le bridge de manière plus automatique avec Ansible. Nous avons gardé les mêmes fichiers de configuration que nos tests précédents cependant, nous avons remplacé les parties à modifier par des variables préfète. Ces variables vont nous permettre de récupérer les informations de notre machine de manière automatique. Par exemple si nous voulons récupérer le nom de notre machine, nous allons utiliser une variable du type {hostname}. Utiliser cette méthode nous permet d'être plus précis dans nos configurations et de donner à notre fichier de configuration ce qu'il attend parfaitement.

Après avoir activé certaine fonctionnalité dans des fichiers de conf, nous avons dû faire quelque manipulation pour générer une clé secrète. Une fois toute nos configuration mis en place nous avons redémarrer nos service Matrix et le bridge étais opérationnelle.

Utilisation du bridge facebook :

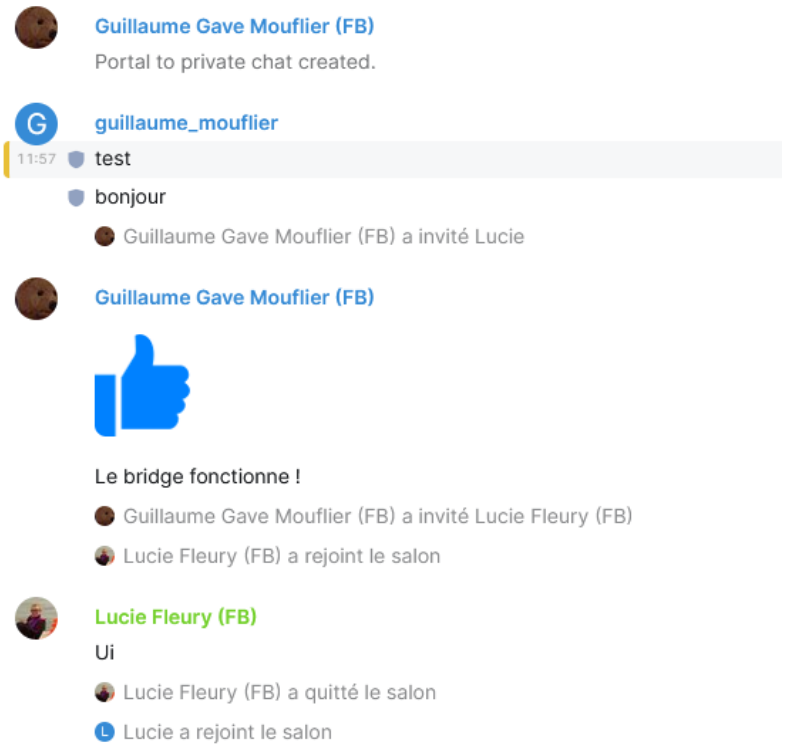
Pour utiliser le bridge facebook nous devons démarrer une conversation avec l'adresse : @facebookbot :azurael.ovh. Cela nous met en contact avec un bot qui vas nous demander de saisir nos informations de connexion. Nous allons devoir renseigner notre adresse mail utilisé pour Facebook ainsi que notre mot de passe.

Voici à quoi ressemble la discussion avec le bot Facebook :



Une fois la connexion réussie par le bot. Nous pouvons discuter d'un environnement Matrix à Messenger.

Sur MATRIX :



Sur Messenger :

Sur le client Matrix, Lucie était connectée avec son compte Facebook et envoyait des messages à Guillaume.

Nous pouvons souligner quelques points à améliorer. Nous avons rencontré quelques problèmes avec l'utilisation de ce bot. Dans un premier temps, Facebook a une politique de sécurité stricte, ainsi lors de notre connexion, le bot Facebook nous a demandé de changer notre mot de passe Facebook. Ce

n'est pas gênant pour l'utilisation du bridge mais il serait bien de désactiver la demande de changement de mot de passe.

Un second point bloquant, lorsque nous nous sommes déconnectés de Facebook avec Matrix, il était impossible de nous reconnecter avec notre compte. Cela veut dire qu'un compte Matrix est limité à une connexion avec le bridge Facebook. En approfondissant nos connaissances et en parcourant les fichiers de configuration il y a possibilité de palier à ces problèmes.

Le problème de connexion peut être pallié par un simple redémarrage de ce service mais il ne s'agit que d'une solution de contournement.

Configuration du Bridge Discord :

Pour créer le bridge en Matrix et Discord, nous avons d'abord pensé à aussi utiliser Puppet via le bridge MC Puppet Discord. Celui-ci est compris dans le playbook et a juste besoin d'être activé.

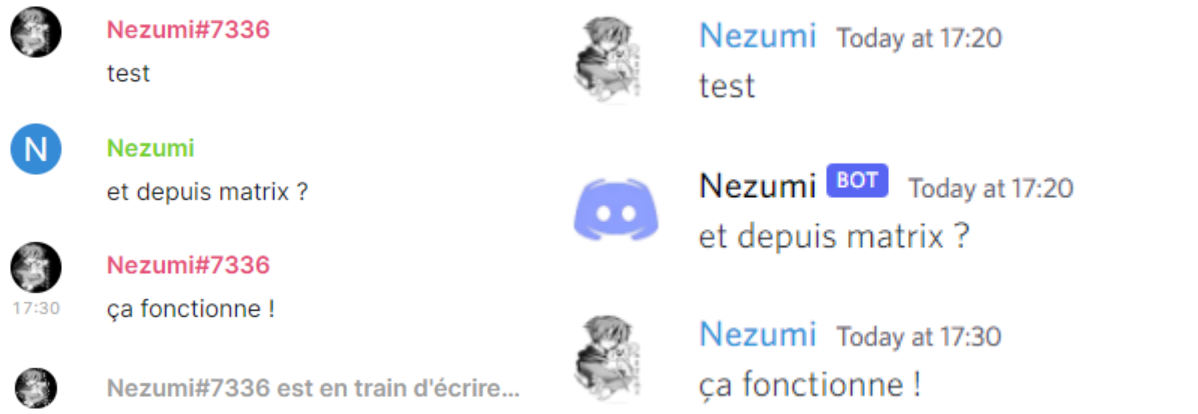
Il faut également créer un bot discord via le portail développer mis à disposition : https://discord.com/developers/applications.

Une fois le bot créé et le token récupéré, il ne manque plus qu'à lier le bot matrix avec le bot discord avec une commande link. Du moins en théorie car une fois les 2 bots liés, il était impossible de mettre en place de bridge. Aucun serveur discord (guild) n'était reconnu par le bot.

Nous avons donc dû utiliser un autre bridge : Appservice Discord. L'avantage étant que nous avons pu réutiliser le bot discord précédemment créé, la mise en place du bridge s'est faite plus rapidement.

De plus, ce bridge possède une option permettant de lier uniquement, et manuellement, les salons voulus (ce qui est très pratique dans notre cas).

Une fois le lien réalisé, tous les messages envoyés dans un salon seront également envoyés dans l'autre :



VIII – Conclusion :

Nous avons pu, au cours de ce projet, déployer un serveur Matrix. Nous avons pu donc toucher à plusieurs technologies différentes, chacune contenant son propre langage de programmation. Nous avons donc pu étendre notre champ de connaissance technique qui nous serons certainement utile dans le futur. Nous avons aussi découvert la difficulté de financement des projets et les différentes solutions possibles.

Ce projet fut également l'occasion de créer quelque chose pour la FIPA, et pas seulement pour nous, et ouvre des possibilités d'amélioration à travers d'autres projets comme un système d'authentification unique (LDAP, CAS, SSO) ou un élargissement du serveur au campus en collaboration avec Minet ou la DISI, ou encore l'ajout de nouveaux bridges afin de créer un système de communication connecté à travers une multitude de messageries différentes.

IX – Sources :

- Le GitHub :

GitHub - spantaleev/matrix-docker-ansible-deploy: Matrix (An open network for secure, decentralized communication) server setup using Ansible and Docker

- Le bridge Facebook :

matrix-docker-ansible-deploy/configuring-playbook-bridge-mautrix-facebook.md at master · spantaleev/matrix-docker-ansible-deploy · GitHub

- Le bridge Discord :

https://github.com/matrix-org/matrix-appservice-discord

https://github.com/spantaleev/matrix-docker-ansible-deploy/blob/master/docs/configuring-playbook-bridge-appservice-discord.md

- Lien vers Element :

https://element.azuraelovh/

- Notre GitLab :

https://gitlabens.imtbs-tsp.eu/guillaume.mouflier/matrix

- Le GitLab du groupe précédent :

https://github.com/maxime0224/project_matrix