h e g

Haute école de gestion
Genève

Étude et mise en œuvre d'un environnement de test configurable sur la blockchain

Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES

par :

Matias SALGUEIRO

Conseiller au travail de Bachelor :

David Billard, enseignant HES

Haute École de Gestion de Genève, le 23 septembre 2022

Haute École de Gestion de Genève (HEG-GE)

Filière IG

Hes·so GENÈVE
Haute Ecole Spécialisée
de Suisse occidentale

Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre Bachelor of Science HES-SO en informatique de gestion.

L'étudiant a envoyé ce document par email à l'adresse remise par son conseiller au travail de Bachelor pour analyse par le logiciel de détection de plagiat URKUND, selon la procédure détaillée à l'URL suivante : https://www.urkund.com .

L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seul< e > le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Genève, le 23.09.2022

Matias SALGUEIRO

Remerciements

J’aimerais tout d’abord remercier Monsieur David Billard qui m’a donné le sujet de mon travail de Bachelor et qui m’a suivi durant la réalisation de celui-ci.

Je remercie également Evan Schleret pour avoir relu mon document et corrigé les éventuelles fautes d’orthographe.

Enfin, je remercie ceux qui m’ont soutenu, mes amis et ma famille, durant ce travail et ces trois années d’école.

Résumé

La technologie de blockchain est un sujet d'actualité qui intéresse beaucoup de monde. Mais peu de nombreux sont ceux capable d'expliquer pourquoi cette technologie est si intéressante et comment elle fonctionne. En effet, la blockchain repose sur beaucoup de concept parfois difficile à appréhender et encore plus difficile expliquer, même pour des personnes du milieu de l'informatique.

Ce document a donc dans un premier temps pour but d'expliquer le plus simplement possible comment fonctionne la technologie de la blockchain. En faisant cela, nous pourront ensuite voir quels usages sont possible et qui pourrait en profiter. Nous pourrons également comprendre les avantages de cette technologie et bien entendu ses inconvénients.

Ensuite, ce document expliquera le fonctionnement de trois plateformes de blockchain très utilisé dans le domaine pour ainsi y voir plus clair sur leur fonctionnement et usage.

Finalement, nous verrons dans ce document comment utiliser l'une de ces trois plateformes présenté, Hyperledger Fabric, en mettant en place un environnement de test basique, utilisable et configurable.

Table des matières

Liste des tableaux

Liste des figures

1. Introduction

Depuis maintenant un peu plus d'une dizaine d'années, on entend de plus en plus parler de blockchain, mais surtout des applications faites avec cette technologie. Ce qui était avant un sujet qui n'intéressait que les personnes baignant dans le monde de l'informatique est, maintenant, devenu un sujet qui intéresse une grande partie de la population et même les gouvernements. En effet, à ce jour, de nombreuses nations telles que la France, la Suisse ou encore la Chine et les États-Unis d'Amérique se sont mis à créer des projets qui reposent sur la technologie de la Blockchain. Il est difficile de nos jours, d'aller sur un média qui n'ai pas parlé au moins une fois de cryptomonnaies ou bien de NFT.

La blockchain s'est fait connaître en grande partie grâce aux cryptomonnaies et notamment la plus célèbre d'entre elle : le Bitcoin. D'autres sont par la suite apparues tels que l'Ether, le Tether ou bien encore le USD Coin, apportant chacune leurs améliorations ou spécificités. Mais la blockchain permet de faire bien plus de choses que de la cryptomonnaie. Elle pourrait, par exemple, permettre de tenir un registre bancaire, médical ou celui d'une bibliothèque. On pourrait aussi tracer l'origine d'un produit ou bien encore certifier l'authenticité d'une œuvre numérique. Les possibilités de la blockchain sont nombreuses et, à l'ère du numérique, elles intéressent beaucoup de particuliers et d'entreprises.

Mais qu'est-ce que c'est la blockchain ? Est-ce facile à utiliser ? Peut-on créer nous même un registre ou bien une cryptomonnaie grâce à cette technologie ? Est-ce sécurisé ? Nous allons tenter de répondre à toutes ces questions dans ce document en faisant tout d'abord une explication en détail de ce qu'est la blockchain, son fonctionnement, sa sécurité, ses diverses utilisations possibles et enfin ses avantages et inconvénients.

Nous allons dans un deuxième temps voir comment il est possible de mettre en place un environnement de blockchain et quels outils sont disponibles pour pouvoir utiliser cette technologie.

Finalement, nous verrons une démarche concrète, étape par étape pour mettre en place un environnement test de blockchain avec Hyperledger Fabric.

2. La Blockchain

Ce chapitre va détailler tout ce qu'il y a à savoir sur la blockchain pour ainsi mieux comprendre son fonctionnement, ses usages et surtout comprendre plus facilement les outils que nous verrons plus tard.

2.1 Définition

> *« Une blockchain est une base de données partagée entre tous les membres d'un réseau dont l'objectif est l'enregistrement chronologique, immuable et sécurisé des transactions réalisées entre les membres de ce réseau. »*
>
> *(Stanislas de Quénetain, 2019)*

La blockchain est une technologie et non un logiciel. Elle permet de mettre en place une base de données qui sera formée de blocs de données reliés les uns aux autres formant ainsi une chaîne. On compare souvent la blockchain à un grand registre partagé.

2.2 Histoire

Les prémisses de la blockchain apparaissent pour la première fois dans la thèse du docteur en informatique et cryptographie David Chaum : « Computer systems established, maintained, and trusted by mutually suspicious groups » publié en 1982. Il y parle alors d'un système de coffre qui contiendrait des données et une fois ceux-ci fermés, ils ne pourraient plus être ouverts sans être détruits.

Plus tard, en 1991, deux chercheurs, Stuart Haber et Wakefield Scott, reprennent les idées de David Chaum en les améliorants. C'est à ce moment-là qu'apparait le terme « bloc ». Leur but est de créer un système où l'horodatage d'un document ne pourrait être modifié. Ils ont par la suite, grâce à l'aide du mathématicien Dave Bayer, réussi à faire en sorte qu'on puisse mettre plusieurs documents dans un seul bloc.

La première vraie utilisation de cette technologie arrivera en 1990 et la société appelée DigiCash fondée par le docteur David Chaum mentionné plus tôt. DigiCash lancera un système de paiement électronique, se rapprochant plus d'un système de cryptomonnaie que de nos services de paiement électronique modernes. Ce système permet de faire des virements intraçables et chiffrés. Malheureusement, David Chaum n'a pas réussi à convaincre les banques de soutenir son projet et DigiCash a déclaré faillite en 1998.

Ce n'est que bien plus tard, en 2008, que la technologie blockchain refait son apparition au grand publique grâce à Satoshi Nakamoto et la cryptomonnaie Bitcoin. Nakamoto reprend l'idée de la blockchain en améliorant encore son design. C'est lui qui met en place un système de « proof-of-work », que nous verrons plus bas, qui permet d'améliorer grandement la stabilité et la sécurité de la blockchain. Le Bitcoin était alors

la première cryptomonnaie à survivre et prendre assez de valeur pour intéresser les investisseurs. Depuis, plusieurs ont suivi comme l'Ether, USD Coin ou le XRP pour n'en citer que quelques-unes.

Bien qu'étant surtout connue pour les cryptomonnaies, la blockchain permet aussi de faire bien d'autres applications que nous verrons par la suite.

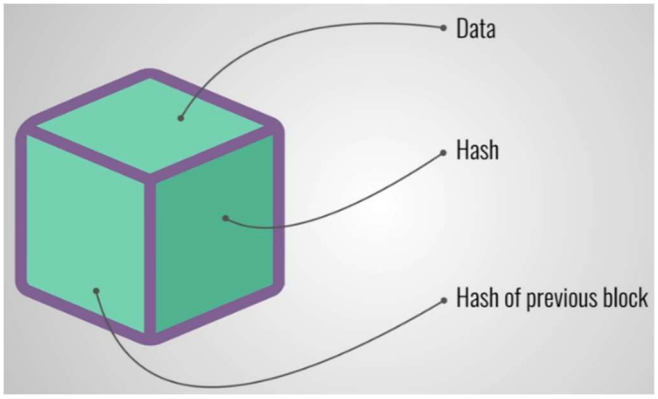
2.3 Fonctionnement

Là où une base de données « classique » fonctionne avec un principe de simple table contenant des données, la blockchain fonctionne sur le principe des blocs également appelés nœuds. Un bloc peut contenir un certain nombre de données ou transactions, qui sont groupées. Celui-ci a une capacité de stockage limité et une fois rempli, il est alors scellé, horodaté et attaché au bloc qui a précédemment été rempli, créant ainsi une chaîne. Lorsque le bloc a été fermé, les données présentes à l'intérieur ne peuvent plus être changées et sont donc immuables. Le but de la blockchain est uniquement de stocker des données sans pouvoir les modifier ou supprimer. Ces blocs, ayant la date à laquelle ils ont été remplis, permettent alors de remonter la chaîne de données dans l'ordre chronologique.

2.3.1 Liaison entre les blocs - le hash

Pour attacher un nœud à un autre, la blockchain utilise le hash. Un hash est une série unique de caractères permettant d'identifier un nœud, comme une empreinte digitale permet d'identifier une personne. Le hash est calculé grâce à une fonction de hachage. Si le hash est calculé deux fois sur un même nœud, la valeur sera toujours la même si le contenu du nœud n'a pas changé. Chaque nœud a donc un hash unique lui correspondant et permettant de l'identifier. Un nœud contient donc des données, son hash et le hash du nœud le précèdent, ce dernier faisant la liaison entre deux nœuds.

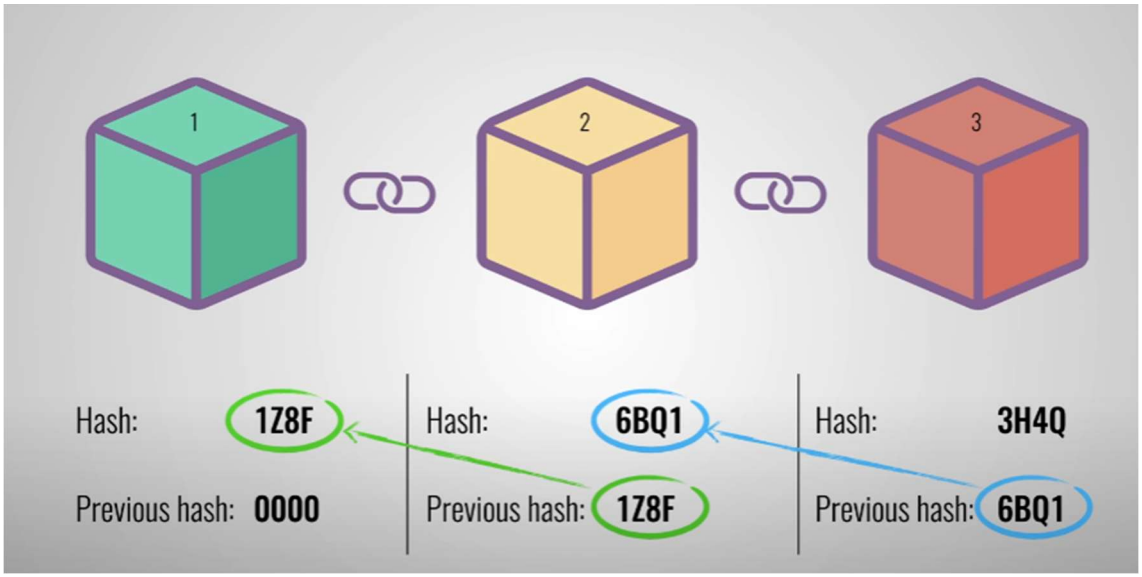
Figure 1 : Bloc de données



(Simply Explained YouTube, 2017)

Le hash d'un nœud est calculé au moment où celui-ci est scellé. Si les données du nœud sont altérées, le hash qui l'identifie changera. Il est alors facile de détecter les modifications. Le seul nœud n'ayant pas de hash d'un nœud le précèdent est celui qu'on nomme « l'originel » ou « genesis block », le premier nœud de la chaîne. Voici a quoi cela ressemblerait :

Figure 2 : Chaîne de blocs



(Simply Explained YouTube, 2017)

Le hash permet donc d'avoir une sécurité sur les blocs en faisant en sorte que si quelqu'un veut modifier les données d'un bloc, il devra alors non seulement recalculer le hash de celui-ci, mais aussi le hash de tous les blocs de la chaîne. Ce qui prendrait énormément de temps. Cependant, avec la puissance des ordinateurs modernes qui ne cesse de croitre, calculer plusieurs milliers, voire millions, de hash est tout à fait faisable en quelques secondes, rendant ainsi cette sécurité inefficace.

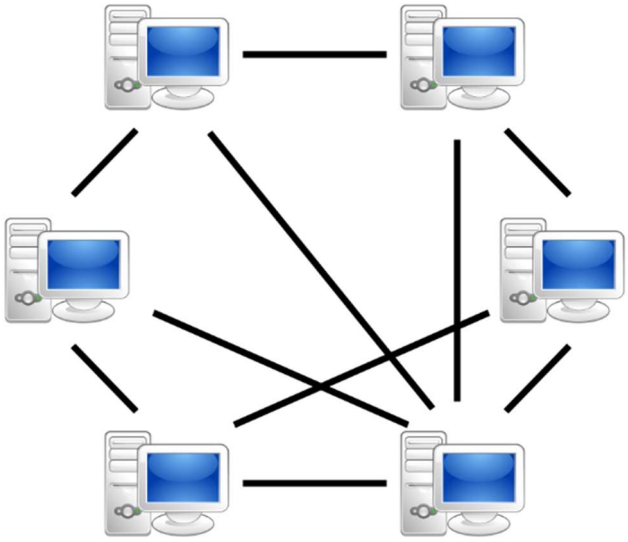
2.3.2 Plus de sécurité – Proof of Work

C'est pour ça qu'il a été ajouté une autre sécurité qui s'appelle le « proof of work ». Cette méthode était à la base utilisée pour limiter les spams par email et a été adaptée à la blockchain. Elle consiste à ralentir volontairement un processus en demandant à l'ordinateur de résoudre un problème qui demande des ressources (CPU). Ainsi, cette méthode permet de rendre bien plus long le fait d'ajouter un bloc à la chaîne. Par exemple, pour le Bitcoin, il faut environ 10 minutes pour faire les calculs nécessaires pour valider un bloc à ajouter à la chaîne. La blockchain se démarque aussi d'une base de données classique par le fait qu'elle est décentralisée. Cela veut dire qu'il n'y a pas d'autorité centrale contrôlant le registre. Pour fonctionner, la blockchain fonctionne donc sur le principe du peer-to-peer.

2.3.3 Décentralisation grâce au réseau peer-to-peer

Le peer-to-peer (P2P) définit un modèle de réseau d'égal à égal entre ordinateurs, où chaque entité est à la fois client et serveur. Cela veut donc dire que chaque ordinateur reçoit, stocke et distribue les données. Le réseau peut être local, c'est-à-dire connecté directement via des câbles, ou bien sans câble via Internet. On appelle un ordinateur dans un réseau peer-to-peer un nœud ou peer.

Figure 3 : Réseau Peer-to-Peer



(Peer-to-Peer Wikipédia, 2022)

Dans un réseau P2P, chaque ordinateur aura alors une copie complète de la base de données et recevra tous les changements faits sur celle-ci. Pour ajouter un bloc il faut alors que celui-ci soit ajouté à toutes les base de données de tous les nœuds du réseau.

Maintenant, imaginons le cas d'un utilisateur A qui est dans un réseau de blockchain. Il crée un nouveau bloc et il veut l'ajouter à la chaîne. Mais avant de faire cela, il faut que les autres utilisateurs du réseau connaissent son bloc et le valide. L'utilisateur A va donc devoir envoyer son bloc, qui aura été scellé et horodaté, à tous les membres du réseau. Mais comment font les utilisateurs du réseau pour vérifier qu'ils ont bien reçu le bloc venant de l'utilisateur A et non pas une copie et que le bloc reçu n'a pas été modifié ?

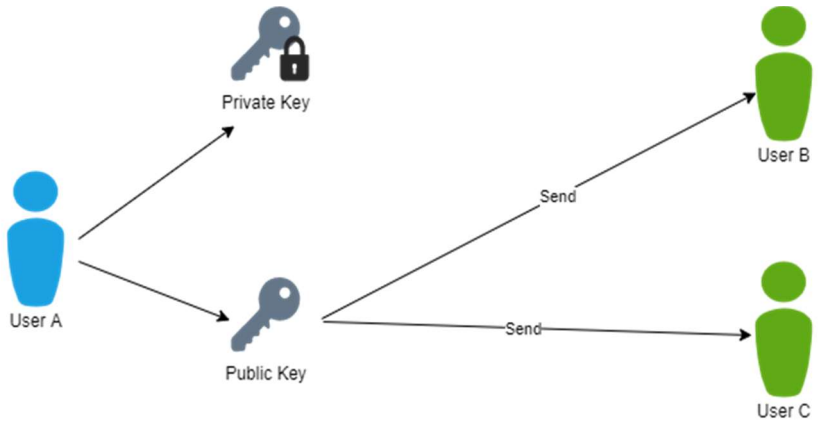
2.3.4 Cryptographie asymétrique et signature numérique

Lorsqu'un bloc est créé, son créateur va le signer numériquement. Une signature numérique est comme la signature sur un document officiel. Si nous reprenons notre exemple plus haut : l'utilisateur A, va signer le bloc qu'il a créé pour attester que c'est bien lui et personne d'autre qui l'a créé. Cela fait aussi en sorte que l'utilisateur A ne puisse dire par la suite que ce n'est pas lui qui a créé ce bloc, mais quelqu'un d'autre

ayant usurpé son identité. Enfin, la signature numérique, permet de garantir l'intégrité du bloc, donc que personne ne l'a modifié. Mais comment cela fonctionne ?

La signature numérique fonctionne sur le principe de clé asymétrique. Chaque utilisateur a deux clés : une clé privée que seul l'utilisateur possède et qui ne doit pas être connu des autres. Une clé publique qui elle peut être transmise à n'importe qui.

Figure 4 : Clé publique et clé privée



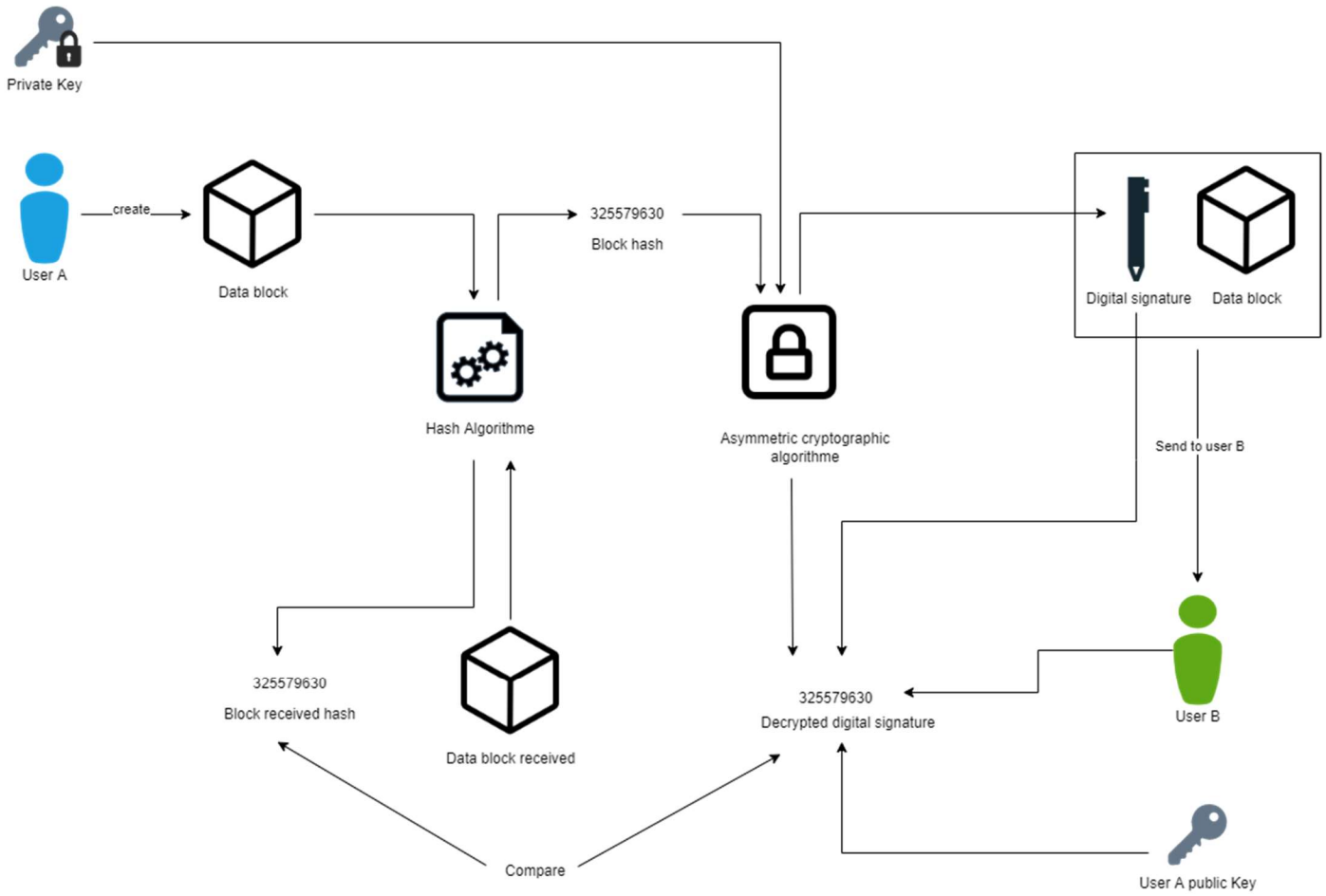
L'utilisateur A va donc créer son bloc. Il va ensuite calculer le hash de son bloc et le chiffrer grâce à sa clé privée et un algorithme de chiffrement asymétrique. Ceci va donc créer la signature numérique du bloc. Une signature numérique est liée à un bloc et est indissociable. Ce n'est pas le bloc qui est chiffré mais seulement le hash de celui-ci.

L'utilisateur A va ensuite envoyer son bloc à tous les autres membres du réseau. Ceux-ci vont alors utiliser la clé publique de l'utilisateur A pour déchiffrer la signature numérique ce qui va donc leur donner le hash du bloc.

Une fois cela fait, les membres du réseau vont alors prendre le bloc reçu et calculer un hash grâce au même algorithme que l'utilisateur A. Ils vont ensuite pouvoir comparer le hash obtenu en déchiffrant la signature numérique au hash créé grâce au bloc reçu. Si les deux correspondent, alors ils sauront que le bloc n'a pas été modifié et ils peuvent alors le valider pour l'ajouter à la chaîne. S'ils ne correspondent pas, les utilisateurs sauront que le bloc a été modifié et ils ne le valideront pas. Il peut aussi y avoir le cas où la clé publique ne fonctionne pas. Cela veut alors dire que le bloc signé reçu n'a pas été signé par l'utilisateur A.

Voici donc à quoi ressemblerait le processus de vérification :

Figure 5 : Schéma cryptographie asymétrique et signature numérique



Le fait que tous les utilisateurs du réseau aient une copie de la base de données fait qu'il est très difficile qu'une personne malveillante puisse altérer les données. En effet, étant donné que chaque membre communique avec tous les autres membres du réseau, il se forme ce qu'on appelle un consensus. Cela veut dire que tous les utilisateurs se mettent d'accord sur les données qui sont valides et celles qui ne le sont pas. Par exemple : si 80% des utilisateurs ont des données qui correspondent, alors ça sera ces données la qui resteront et les autres ne seront pas prises en compte jusqu'à ce qu'elles soient identiques. Voilà pourquoi un bloc doit être envoyé à tous les membres et vérifié par chacun. Ce consensus ce fait grâce à des protocoles de consensus comme le proof-of-work que nous avons vu plus tôt.

2.3.5 Résumé

La blockchain est une technologie permettant de mettre en place des bases de données qui n'ont pas d'autorité centrale. Elle fonctionne sur un système de blocs de données qui sont ensuite attachés les uns aux autres formant ainsi une chaîne. La connexion entre

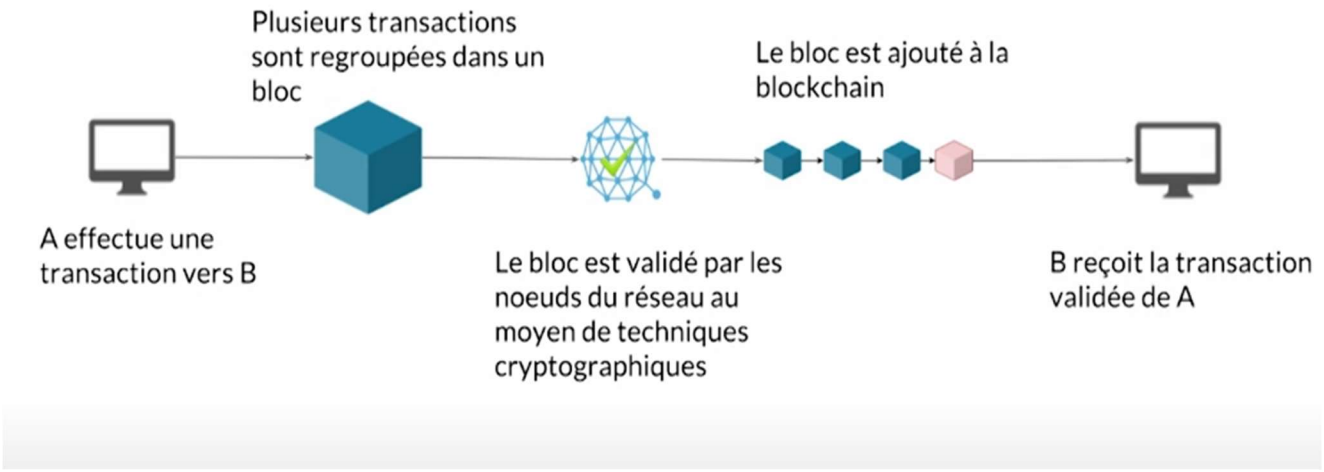
les blocs se fait grâce au hash de ceux-ci, chaque bloc ayant la référence au hash du bloc précédent.

Comme elle n'a pas de serveur central, la blockchain fonctionne sur un réseau peer-to-peer où chaque utilisateur de ce réseau est à la fois client et serveur. Chaque utilisateur possède donc la base de données au complet et reçoit, stocke et distribue les données.

Il faut aussi rappeler que toutes les méthodes expliquées jusqu'à maintenant ne sont pas faites manuellement par un utilisateur, mais automatiquement par l'application de blockchain utilisée.

Les explications sur le fonctionnement de la blockchain faites sur ce document ne sont pas exhaustives. Chaque application de blockchain a ses subtilités. Mais le fonctionnement général reste le même.

Figure 6 : Schéma montrant le parcours d'une transaction dans la blockchain



(Vincent Segouin, 2018)

2.4 Est-ce que la blockchain est sécurisée ?

Lorsque l'on parle d'application de blockchain, comme le Bitcoin, on la présente presque toujours comme étant fiable sécurisé et ne pouvant être piratée. Mais est-ce le cas ? Revenons sur les points essentiels dont nous avons parlé et qui permettent de rendre la blockchain plus sûre.

2.4.1 Les méthodes pour sécuriser la blockchain

Tout d'abord, la décentralisation de la blockchain permet d'avoir une redondance de la base de données. C'est-à-dire qu'il y aura plusieurs copies à plusieurs endroits différents. Ceci permet d'éviter d'avoir ce qu'on appelle un point de défaillance unique. Un point de défaillance unique est un point d'un système qui, s'il fait défaut, entrainera

l'arrêt complet du système, ce qui est souvent le cas pour une entreprise où un seul serveur contient toute la base de données. Ce défaut peut être causé par plusieurs évènements : incendie, panne, erreur humaine, etc. Dans le cas d'un incendie, le système pourrait même être détruit, entrainant ainsi la perte des données. Avec la décentralisation, on réduit ce problème presque à zéro. Pour qu'un système décentralisé tombe, il faudrait que l'entièreté des ordinateurs du réseau tombe en même temps pour que le réseau soit hors ligne. De plus, le fait que tous les ordinateurs du réseau aient une copie identique de la base de données rend très difficile le fait de pouvoir corrompre les données. Comme nous l'avons vu, le réseau utilise un protocole de consensus, ce qui fait qu'il faudrait que plus de la moitié des membres du réseau soit corrompus pour que la base de données entière le soit.

Ensuite, le hash des blocs et le fait que ceux-ci soient rattachés les uns aux autres, rend très difficile le fait de pouvoir les modifier ou supprimer sans être détecté. En effet, étant donné qu'un bloc possède un hash qui est calculé au moment où il est fermé, et que celui-ci a le hash du bloc précédent, il faudrait modifier tous les blocs de la chaîne pour que le changement ne soit pas rejeté. Ce qui demanderait des ressources énormes pour pouvoir recalculer tous les hash sans que personne ne s'en aperçoive. Vient s'ajouter à cela, le proof of work qui va ralentir la validation d'un bloc ce qui rendrait l'opération de corruption d'un bloc dans la chaîne incroyablement longue. De plus, il ne faudrait pas seulement faire ça sur un membre du réseau, mais sur au moins 51% des membres, l'opération devient alors quasi impossible.

Finalement, il y a la signature numérique d'un bloc qui permet de s'assurer :

- Que le bloc n'ait pas été modifié entre le moment où il a été scellé et le moment où il a été reçu par les autres membres du réseau.
- Qu'il ait bien été envoyé par le même membre qui a créé ce bloc.

Cependant, il faut souligner que toutes ces méthodes de sécurisation de la blockchain ne sont pas faites par défaut. En effet, étant donné que la blockchain est une technologie et non pas une application, rien ne garantit qu'un projet de blockchain ai toutes ces sécurités incluses.

2.4.2 Problèmes du proof of work

Bien que très efficace pour sécuriser la blockchain, le proof of work (POW) a trois gros problèmes. Pour les comprendre, revenons sur le fonctionnement de cette méthode et comment elle est appliquée au Bitcoin. Le Bitcoin fonctionne ainsi : lorsqu'un bloc est créé, il est envoyé à tous les membres du réseau. Une fois que le bloc a été authentifié grâce à la signature numérique, les utilisateurs vont alors l'ajouter à leur copie de la

blockchain. Pour ce faire, ils vont devoir résoudre un « puzzle » algorithmique demandant une grande puissance de calcul aux ordinateurs (la méthode du proof of work). L'ordinateur du réseau Bitcoin qui résout en premier le puzzle de vérification reçoit alors en récompense un Bitcoin. C'est ce qu'on appelle le « Bitcoin Mining ».

Premier problème : les mining pools. Étant donné qu'il faut une grande puissance de calcul, demandant des composants puissant et cher, pour résoudre les puzzles de vérification, certaines personnes ont eu l'idée de regrouper leurs ordinateurs dans un seul endroit pour allier leur puissance pour créer des pools[1] de nœuds. Cela permet ainsi d'avoir plus de chance d'être celui qui est récompensé par le système.

Mais cette méthode va à l'encontre de l'idée de la décentralisation. Plutôt que d'avoir des ordinateurs éparpillés tout autour du globe, nous nous retrouvons maintenant avec de très grand hangar accueillant plusieurs milliers de membres du réseau. Il suffirait alors que plusieurs de ces hangars soit compromis pour avoir une grande partie du réseau Bitcoin qui tomberait entre de mauvaises mains.

Figure 7 : Ferme à Bitcoin

(cnbc.com, 2021)

Deuxième problème : le gaspillage de ressources matérielles et d'énergie. Comme dit précédemment, il faut une grande puissance de calcul pour pouvoir résoudre les puzzles

[1] Qu'est-ce qu'un mining pool et quel est son fonctionnement ?, 2022. Bitpanda [en ligne]

algorithmiques du Bitcoin. Ce qui fait que les composants puissants sont convoités par les mineurs et leurs fermes à bitcoin qui achète en masse des composants d'ordinateur très puissants, faisant ainsi augmenter les prix sur les marchés. Tous ces composants, étant très puissants, demandent beaucoup d'énergie pour fonctionner. De plus, la méthode proof of work récompense un seul nœud du réseau pour la résolution du problème, mais tous les autres nœuds doivent aussi réussir le puzzle algorithmique. Ce qui fait que beaucoup d'énergie doit être dépensée pour ajouter un bloc à la chaîne.

2.4.3 Proof of Stake, l'alternative au proof of work ?

En réponse aux problèmes de la méthode proof of work est apparue une autre méthode pour tenter de la remplacer : le proof of state (POS). Cette méthode est le plus souvent utilisée dans les cryptomonnaies. Le POS fonctionne de la manière suivante : au lieu d'avoir tous les membres du réseau qui valident un nouveau bloc, seuls quelques utilisateurs sont choisis par le système pour effectuer le travail de validation. Ils seront ensuite récompensés. On appelle les utilisateurs choisis les « validators ». Mais pas n'importe quel utilisateur peut être choisi. Pour être éligible à la validation de nouveaux blocs, un utilisateur doit offrir en échange de sa participation une contrepartie en coins. On peut comparer ça à un gage fait auprès d'une banque à laquelle on empreinte de l'argent. Si le POS était utilisé pour le Bitcoin, il faudrait alors que l'utilisateur donne en gage quelques-uns de ses Bitcoins. Plus le gage donné est grand, plus il y a de chances qu'on soit choisi par le système pour valider un bloc. Une fois le bloc validé par les validators, il est alors ajouté à la chaîne. Si un utilisateur valide un bloc « frauduleux », il sera alors pénalisé et se fera retirer des coins.

Cette méthode permettrait de résoudre les deux problèmes du proof of work cités plus tôt. En effet, étant donné que ce n'est plus le premier à avoir résolu le puzzle algorithmique qui reçoit une récompense, la puissance du matériel ne sert plus à rien. Cela veut donc dire que les pools de nœuds ne sont plus nécessaires pour recevoir beaucoup de récompenses. Le problème de l'énergie est également résolu, car il n'y a plus tout le réseau qui doit valider un bloc, mais seulement quelques membres. L'Ethereum vient de passer récemment au POS et déclare que sa consommation d'énergie va baisser de 99.5 %.

2.4.4 Applications de la blockchain

> *« (…) blockchain immutability is called into question not by exploiting cryptographic vulnerabilities but instead by subverting the properties of a blockchain's implementation, networking, or consensus protocol. A subset of a blockchain's participants can garner excessive, centralized control over the entire system. (…) »*
>
> *(DARPA, 2022)*

La question à se poser n'est donc pas « est-ce que la blockchain est sécurisée ?» mais plutôt « est-ce que cette application de blockchain est sécurisée ? ». Ce n'est pas la technologie qui est sécurisée mais la mise en place lorsque on l'utilise dans une application. Ce qui veut dire que la sécurité est donc la même que toutes les autres applications existantes. Si le code de l'application blockchain que l'on est en train de faire a des failles, alors la blockchain aura forcément des failles qui pourront être exploitées par des personnes malveillantes.

2.5 Usages de la blockchain

Maintenant que nous connaissons bien mieux la blockchain, nous pouvons nous demander quelles seraient les différentes utilisations possibles. Il faut garder en tête que la blockchain, étant utilisée pour créer une base de données, pourrait servir à n'importe quel service nécessitant des transferts et stockages de données.

2.5.1 Cryptomonnaie

L'utilisation la plus connue de la blockchain est bien évidement la cryptomonnaie. Selon Lyle Daly, rédacteur chez The Motley Fool, il y aurait en 2022 plus de 12'000 cryptomonnaies en circulation. Mais à quoi sert une cryptomonnaie ?

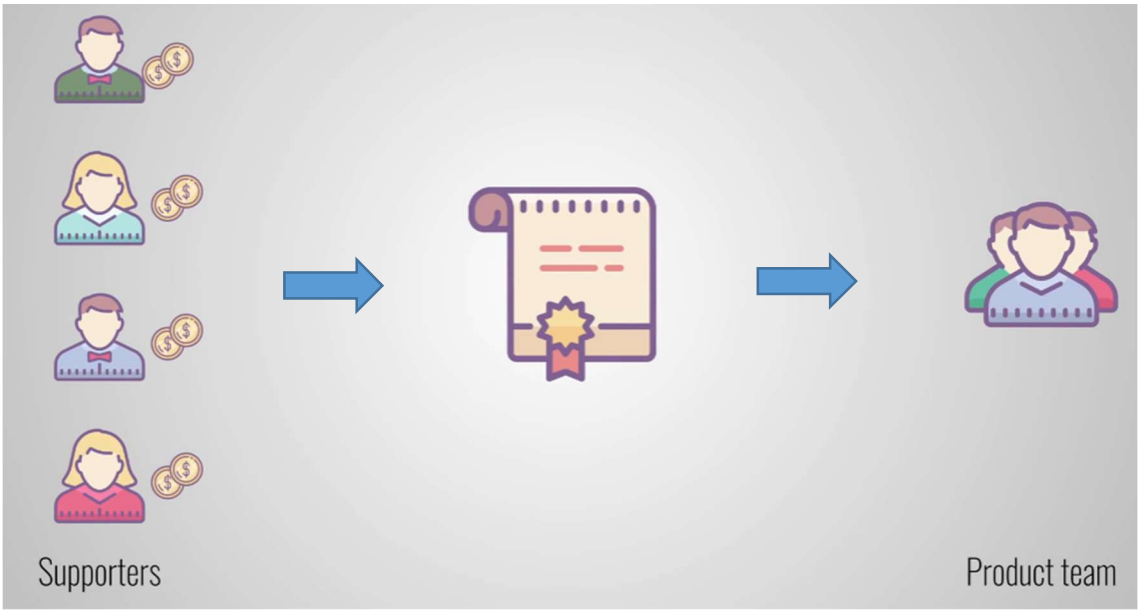
Les systèmes de cryptomonnaie existant sont souvent comparés à une devise comme l'euro ou le dollar, mais ils se rapprochent plus de lieu d'échange décentralisé numérique. Un peu comme une banque, mais sans banquier, et sans entité qui prend de l'argent sur les transactions financières, le tout à travers internet. Une personne aura un portemonnaie virtuel et pourra faire des échanges avec d'autres personnes possédant également un portemonnaie virtuel. Tous cela dans l'anonymat le plus total, car contrairement à une banque, pas besoin de pièce d'identité pour pouvoir utiliser le système. C'est pourquoi l'utilisation de cryptomonnaie est souvent lié à des activités illégales. Bien sûr cela ne représente pas l'entièreté des cryptomonnaies. L'anonymat et la confidentialité était donc l'intérêt principal des premières cryptomonnaies. Mais aujourd'hui, beaucoup d'entre elles sont souvent utilisées de manière purement spéculative.

2.5.2 Smart contracts

Le smart contract est programme informatique qui est stocké dans un bloc d'une blockchain et qui est l'équivalent d'un contrat mais virtuel. Ce programme informatique fonctionne comme un intermédiaire entre deux parties.

Prenons l'exemple d'un site de financement participatif : un groupe de personnes – groupe A veut investir dans le projet de Monsieur B. L'intermédiaire serait le site sur lequel le financement a eu lieu. Il faudrait alors que le groupe A et monsieur B fassent tous les deux confiances dans le site de CrowdFunding ce qui peut poser quelques problèmes. Que se passe-t-il si le site ferme ? Où va l'argent ? Comment faire confiance à un site ? C'est là que le smart contract est utile. Il permet de remplacer l'intermédiaire par un simple programme informatique qui sera ensuite stocké dans une blockchain. Le programme fonctionne selon des paramètres définis au moment de la création de celui-ci.

Figure 8 : Smart Contract



(Simply Explained YouTube, 2017)

Dans notre exemple, le groupe A va mettre l'argent pour le projet de Monsieur B sur le smart contract. Si le projet a bien été complètement financé et que tous les membres du réseau valident que les objectifs financiers du projet ont bien été atteints, alors le smart contract versera l'argent à monsieur B. Si ce n'est pas le cas, l'argent sera alors remboursé au groupe A. L'intérêt d'une telle application est qu'il n'y a pas d'intermédiaire, comme une banque ou un site. Personne ne contrôle l'argent, car c'est un programme qui va le faire. Etant donné que le contrat se trouve dans un bloc de la blockchain, il est alors immuable et ne peut donc pas être modifié. Il est aussi bien sûr distribué à tous les membres du réseau blockchain qui le valideront ou pas.

Il y a bien sûr des gros risques sur cette utilisation. Comme dit dans le paragraphe sur la sécurité : *« Ce n'est pas la technologie qui est sécurisée mais la mise en place lorsque on l'utilise dans une application. ».* Ce qui veut dire que le programme servant de smart

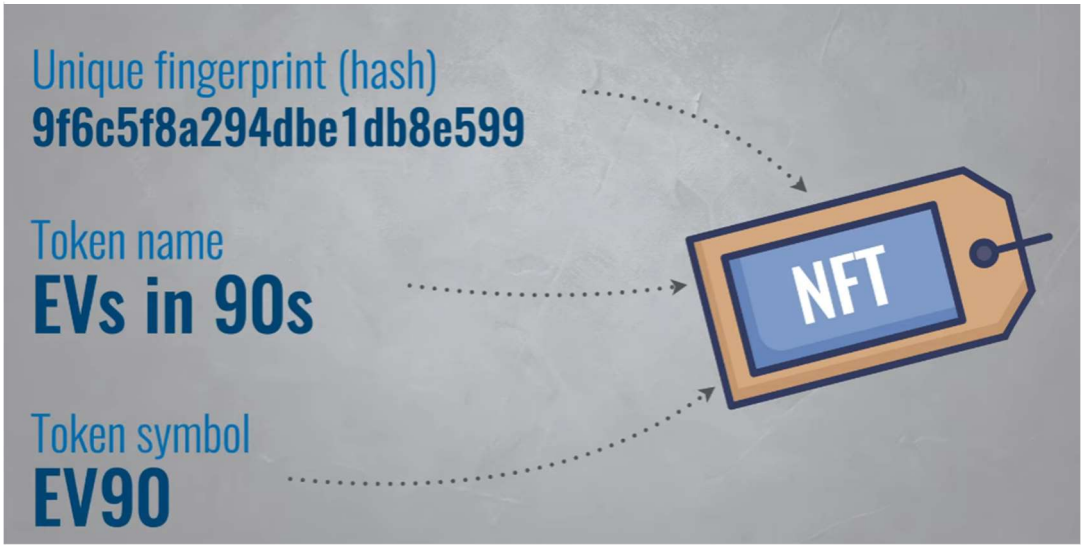
contract peut être corrompu s'il n'est pas bien sécurisé. C'est d'autant plus dangereux ici car les smart contracts gèrent des sommes d'argent.

2.5.3 Non-Fungible Token (NFT)

Une des utilisations qui fait le plus de bruit depuis peu est le « Non-Fungible Token » ou « NFT » qui se traduit en français par jeton non fongible. Fongible se dit d'un bien pouvant être échangé contre un autre bien de même valeur et quantité. Par exemple la monnaie ou le riz sont des biens fongibles, un dollar équivalant à un autre dollar. Non fongible veut alors dire que ce bien est unique et ne peut donc être échangé contre un jeton de même valeur, ce qui ne veut pas dire qu'il ne peut pas être vendu. Les NFT peuvent représenter tout ce qui est numérique : une œuvre d'art, un ticket de concert ou bien encore un certificat de propriété. Le NFT n'est pas le bien numérique en lui-même mais plutôt un titre de propriété consignée dans un registre décentralisé, la blockchain.

Un NFT permet de garantir qu'une personne possède bien une copie originale d'un bien. Prenons l'exemple d'un ticket de concert : vous achetez un ticket sur un site en ligne et celui-ci vous donne alors un NFT de ce ticket. Ce NFT aura un hash du ticket permettant de l'identifier, un Token name, et un Token. Ce Token est alors stocké dans un bloc de la blockchain ayant votre signature numérique, attestant donc que vous êtes le seul et unique propriétaire. Ainsi, même si une personne venait à avoir une copie de ce ticket, il serait invalide car il ne possède pas l'original.

Figure 9 : Contenu NFT



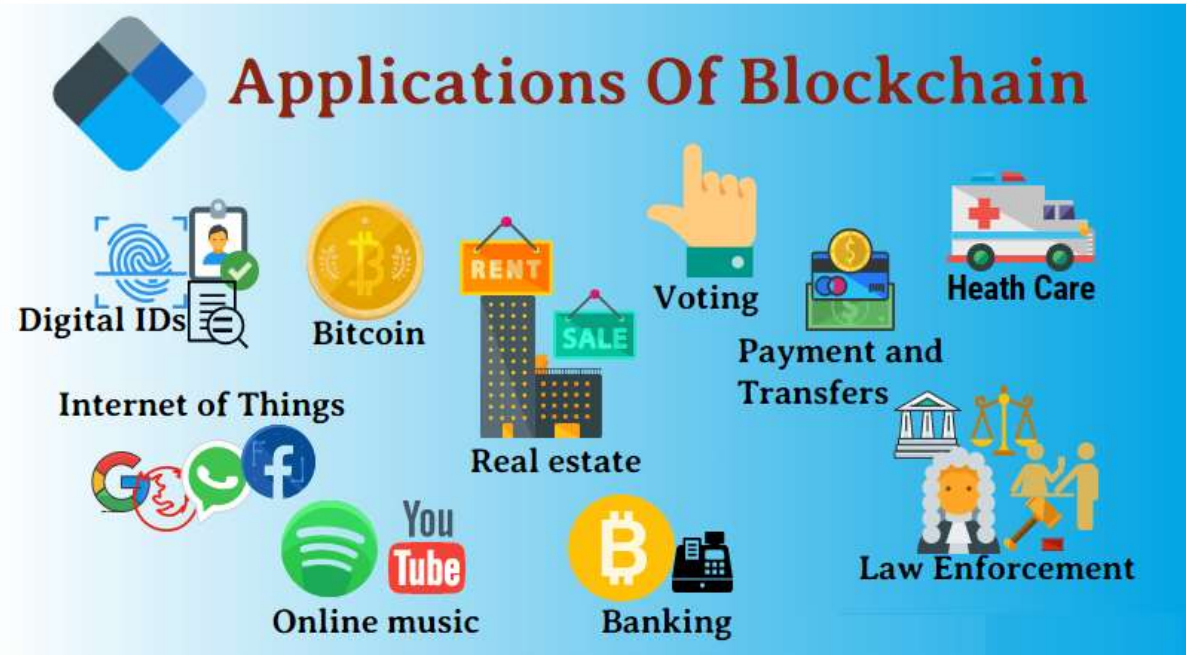
(Simply Explained YouTube, 2021)

Les NFT sont souvent liés à des smart contracts. Ce smart contract pourrait par exemple faire en sorte que lorsqu'un NFT d'une œuvre d'art est vendue, des royalties soient versées au créateur. On voit alors l'utilité d'une telle utilisation pour des entreprises et

bien sur les artistes. Mais celle-ci est souvent éclipsée par les médias qui ne parlent souvent que de ventes de NFT ne servant que dans un but spéculatif.

2.5.4 Autres utilisations possibles

Figure 10 : Applications de la Blockchain



(medium.com, 2020)

Comme dit précédemment, la blockchain pourrait servir dans toutes les applications qui ont besoin d'avoir une base de données. Nous avons vu les trois utilisations les plus populaires dans le les applications de blockchain. Mais tous les domaines peuvent en bénéficier : documents d'identité, registres médicaux, votations etc. Certains experts estiment même que la prochaine évolution d'internet se fera grâce à la blockchain, en mettant en place un internet totalement décentralisé et dont les données seraient stockées dans des blocs.

Si nous prenons deux exemples : IBM propose une solution basée sur la blockchain pour améliorer la chaîne d'approvisionnement d'une entreprise. Le but de cette solution est de remplacer les bases de données des différentes entreprises pour créer un grand réseau décentralisé contenant toutes les informations nécessaires au bon fonctionnement de l'approvisionnement. Le but étant de supprimer les erreurs humaines et accélérer les processus qui prennent bien souvent trop de temps.

Un autre exemple : en 2018 aux États-Unis d'Amérique, lors des élections de mi-mandat, l'état de Virginie de l'ouest a mis en place un projet test de votation sur une plateforme basée sur la blockchain. Ce projet avait pour but de mettre en place une plateforme sur

pour pouvoir voter sans possibilité de fraude. Il fût une réussite, mais beaucoup d'experts du monde informatique étaient mitigés sur les résultats affirmant qu'une telle plateforme serait bien trop compliquée à maintenir et à sécuriser sur une plus grande utilisation.

Tous les exemples montrés dans ce chapitre montrent bien la diversité possible dans l'utilisation de la blockchain. Il faut cependant garder en tête que plus le réseau créé sera grand, plus grands seront les menaces et risques encourus. De plus, il est possible de créer soit des applications de blockchain publique ou privée. Publique veut dire que n'importe qui peut rejoindre le réseau et devenir un nœud. Privée veut dire qu'un membre doit d'abord être validé par les autres membres du réseau pour pouvoir participer à la blockchain. La sécurité est souvent moins mise à rude épreuve dans des réseaux privés car tous les membres sont de confiance.

2.6 Avantages et inconvénients

Résumons donc les avantages et inconvénients d'utiliser des applications utilisant la technologie de la blockchain :

2.6.1 Avantages

- Erreurs humaines réduites drastiquement grâce au système de vérification basé sur le consensus.
- Réduction des coûts car il n'y a plus besoin d'avoir un humain pour vérifier une transaction, un document, etc.
- Sécurité des données accrue grâce à la décentralisation. La redondance des données enlève le risque de point de défaillance unique.
- Efficacité accrue étant donné que tout est transmis, vérifié et validé automatiquement par les applications. Il n'y a pas d'autorité centrale qui peut ralentir un processus ou l'annuler.
- Confidentialité totale lors des transactions sur la blockchain. Aucune information personnelle étant enregistrée sur les blocs, seulement la signature numérique.
- Sécurité des transactions garantie grâce aux méthodes d'immuabilité et d'intégrité.
- La plupart des blockchains, comme le Bitcoin, ont leur code source qui est visible par tous. Les programmes open source donnent une transparence sur le fonctionnement de l'application.

- Facilement accessible à toutes et tous. Une simple connexion à internet suffit.

2.6.2 Inconvénients

- Coût technologique et énergétique élevé. Les millions de nœud d'un réseau de blockchain nécessitent tous de l'énergie pour pouvoir effectuer le grand nombre de calculs nécessaires à l'ajout d'un bloc à la chaîne. Selon un rapport de l'université de Cambridge, le réseau du Bitcoin consommerait à lui seul plus que ce que les Philippines consomment en électricité sur une année.

- Le système de création proof of work ralentit énormément la création de nouveaux blocs. Rappelons qu'il faut 10 minutes pour créer un bloc sur la chaîne du Bitcoin faisant ainsi tomber ses transactions par secondes (TPS) a 7, là où Visa est capable d'en faire 65'000. D'autres blockchains sont bien entendu plus efficaces comme Solana qui arrive à 8000 TPS mais la majorité des blockchains en sont bien loin.

- Etant donné le caractère confidentiel très fort, beaucoup d'application de blockchain sont utilisées pour des activités illégales.

- Une régulation faible, voire inexistante dans certains pays.

- Sécurité qui est dépendante de l'application et du langage de programmation utilisée.

3. Trois plateformes de blockchain disponible

Il faut tout d'abord faire la distinction entre plateforme de blockchain et application. Une plateforme blockchain est la fondation sur laquelle les applications sont construites, comme un langage de programmation. Les applications de blockchain sont donc l'utilisation d'une plateforme pour créer, par exemple, des NFT, cryptomonnaies, smart contracts, etc. Si nous faisons une comparaison : internet est la plateforme et les sites web sont les applications.

Il faut également différencier les blockchains « permissionless » et « permissioned ». Permissionless veut dire qu'il n'y a pas besoin de permission pour entrer dans le réseau, valider les blocs, etc. On parle alors souvent de blockchain publique. Permissioned veut dire le contraire, il faut être validé par les membres avant de pouvoir entrer dans le réseau. On parle alors de blockchain privée.

Nous allons voir dans ce chapitre, trois plateformes de blockchain disponible, leurs fonctionnements ainsi que leurs applications possibles. Nous allons également voir les spécificités de chacune ainsi que leurs avantages et inconvénients.

3.1 Ethereum

Ethereum est l'une des plateformes de blockchain les plus anciennes et aussi l'une des plus connues. Elle a été conçue en 2013 et rendue publique en juillet 2015. C'est une blockchain publique et open-source ayant comme fonctionnalité principale les smart contracts. Au départ, elle utilisait comme protocole de consensus, le proof of work, mais la plateforme a récemment changé pour une utilisation du proof of stake[2]. Ethereum utilise plusieurs langages de programmation dont le plus utilisé est Solidity[3].

Beaucoup d'utilisations sont possibles avec Ethereum. Tout d'abord les cryptomonnaies. Celle de base d'Ethereum se nomme l'Ether (ETH). La fonctionnalité principale de cette plateforme de blockchain est, comme dit précédemment, les smart contracts. N'importe qui peut en créer un, grâce à Ethereum. C'est par ailleurs Ethereum qui a lancé une norme dans les smart contracts appelé ERC20, qui permet de rendre n'importe quel smart contract interchangeable avec un autre smart contract, peu importe la blockchain. Ethereum permet bien évidemment de créer des NFT, qui sont ensuite échangeables sur diverses plateformes comme OpenSea ou bien Rarible.

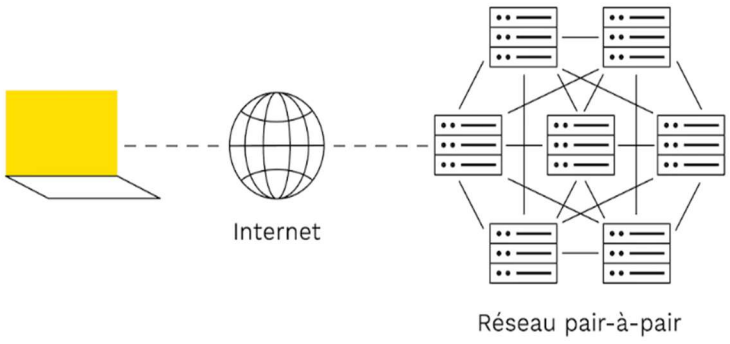
[2] FABRION, Maxence, 2022. The Merge : Ethereum est officiellement entré dans une nouvelle ère.
[3] KAUR, Rashmeet, 2021. Which programming language is used in Ethereum Blockchain ?

Il y a sur la blockchain Ethereum deux types de comptes : le premier est le compte utilisateur « classique », également appelé « externally-owned account ». Ce compte sert uniquement à recevoir ou envoyer la cryptomonnaie Ether. Le deuxième type de compte est celui des smart contracts. Ce compte contrat est associé à un smart contract qui sera déployé sur la blockchain. Les comptes contrat peuvent également recevoir ou envoyer de l'ETH mais toutes ces transactions sont faites par le programme. Ethereum a également un système qui permet d'être soit un « full node » soit un « light node ». Un ordinateur full node va sauvegarder l'entièreté de la chaîne et va également valider les nouveaux nœuds à ajouter. Tandis qu'un membre light node ne va faire qu'utiliser la blockchain, sans avoir de copie de celle-ci.

Il y a sur Ethereum ce qu'on appelle des frais de transactions. Imaginons par exemple que nous sommes un light node et que nous voulons créer un smart contract. On va donc coder le programme et nous allons ensuite devoir l'envoyer sur le réseau pour que les full nodes le mettent sur la chaîne. Mais pour ce faire, il faut associer un certain montant d'ETH à ce smart contract pour que les full nodes le prennent en charge et l'ajoute au registre. Il y a un montant de frais de transaction minimum qui est calculé lorsqu'une transaction est faite. Ces frais seront compris dans la récompense du validator qui va traiter l'opération. Si aucuns frais n'est associé à une transaction, elle ne sera jamais traitée. Ces frais ont été mis en place pour éviter la surcharge des validators avec des transactions inutiles. De plus, bien qu'il y ait un seuil minimum pour les frais, rien n'interdit de mettre plus d'ETH. Plus les frais d'une transaction seront importants, plus la transaction sera traitée rapidement.

On appelle les applications faites grâce à la blockchain des DApps[4]. Ces DApps sont des sites internet, mais sur la blockchain. N'importe quel type d'application peut tourner sur la blockchain.

Figure 11 : Application décentralisé



(bitpanda.com, 2022)

[4] Qu'est-ce qu'une DApp ? 2022. Bitpanda [en ligne]

On peut voir sur le site d'Ethereum une variété d'applications qui vont de l'assurance en passant par le portfolio ou bien encore des sites de CrowdFunding.

Un des problèmes de Ethereum, comme toutes les blockchains ayant comme fonctionnalité de base les smart contracts, est la programmation. Le problème est qu'une fois un smart contract codé, il est impossible de le modifier ou bien de le supprimer. Il ne faut alors pas faire d'erreur dans le programme pour éviter des failles de sécurité.

Ethereum ne peut pas être permissioned et donc privé. Cependant, il existe divers projets prenant comme base l'Ethereum permettant de créer une blockchain privée qui est compatible.

Ethereum suit actuellement une roadmap[5] de développement qui va permettre d'améliorer considérablement sa stabilité, sa vitesse et réduire ses frais de transactions.

3.2 EOSIO

EOSIO est une plateforme de blockchain lancée en 2017. Elle a, comme Ethereum, une fonctionnalité de smart contract. Mais ses points forts résident dans le nombre de transactions par seconde (TPS) possible ainsi que l'élimination de frais de transactions. Comme nous avons pu le voir au point 2.7.2, nombreuses blockchains ont un nombre de TPS relativement bas. Les applications de blockchain utilisant EOSIO sont capables d'atteindre 10'000 TPS, ce qui en fait l'une des plus rapides. EOSIO a fait en sorte, grâce à son architecture, qu'il n'y ait pas de frais de transactions.

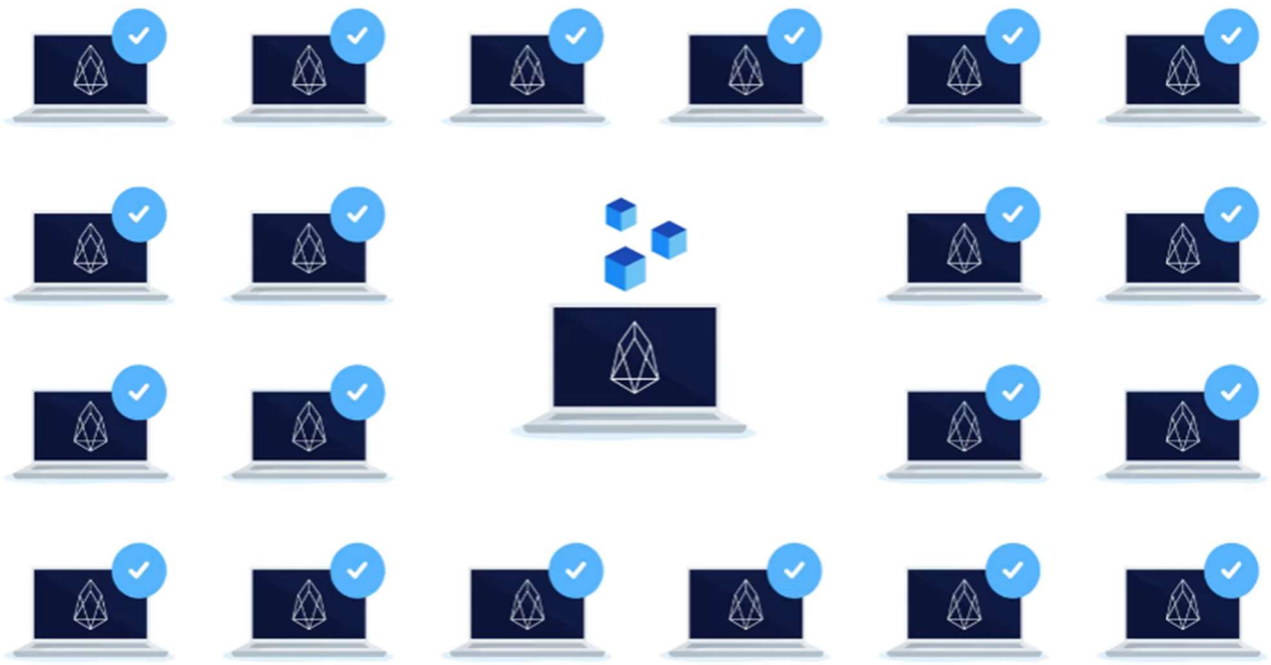
En effet, EOSIO utilise une variante du POS qui s'appelle le « delegated proof of stake » (DPOS). Le système est le même que POS mais, au lieu que ce soit un algorithme qui choisisse les validators, ce seront les utilisateurs qui voteront pour ceux qui créeront et valideront les blocs. Ceux qui sont choisis sont appelés les « block producers » (BP). Leur nombre peut varier entre 21 et 101 selon les besoins. Les BP vont, à tour de rôle, valider les transactions et les ajouter à la chaîne. Les autres BP vont valider le travail du validator et vont, eux aussi, ajouter le block à la chaîne. La nouvelle chaîne sera ensuite distribuée au reste du réseau non BP.

Les block producers sont récompensés par la cryptomonnaie de EOSIO, le EOS, pour la création et la validation des blocks. Pour faire en sorte qu'un grand nombre de membres se présentent à l'élection, il y a également une récompense pour ceux qui sont

[5] CNBCTV18, 2022. « 5 Key events that must occur for Ethereum to hit 100K transactions per second. ».

élus. Mais DPOS n'est pas la seule raison pour laquelle il n'y a pas de frais de transaction sur EOSIO.

Figure 12 : Delegated Proof-of-Work



(Coinbase YouTube, 2019)

Pour comprendre comment EOSIO fait pour que ses utilisateurs n'aient pas de frais de transactions, il faut d'abord expliquer comment fonctionne la cryptomonnaie d'une blockchain. Toutes les blockchains permissionless ont une monnaie virtuelle (coins) qui leur est associée. Cette monnaie permet, entre autres, de récompenser les mineurs ou validators qui font en sorte que le réseau fonctionne. Mais à la création d'une blockchain, il n'y a pas encore de coins, c'est pourquoi les organisations qui mettent en place les projets de blockchain font ce qu'on appelle un « initial coin offering » ou ICO. On peut comparer cette opération à une entreprise qui lève des fonds pour financer un projet. Les investisseurs achètent donc à ce moment-là les premiers coins de la blockchain. Ensuite, ça sera le système qui va continuer à créer les coins pour récompenser les mineurs ou validators. Il y aussi les frais de transaction qui vont faire partie de la récompense des mineurs.

Dans une blockchain comme le Bitcoin, les mineurs vont revendre petit à petit leurs coins contre de l'argent, comme le dollar ou l'euro, ou bien acheter directement avec leurs coins du matériel pour continuer leurs opérations. En faisant ça, ils vont alors augmenter le nombre de coins en circulation. La blockchain va alors s'auto réguler. Dans le cas de EOSIO, la plateforme a choisi d'enlever les frais de transaction et ne laisser que la récompense de validation et la récompense d'élection en EOS pour les blocks

producers. La plateforme prend donc en charge les frais de transaction pour attirer le plus de personnes possibles sur son réseau.

Il y a cependant un problème avec cette méthode. En effet, comme nous l'avons vu plus tôt, le fait de mettre plus d'ETH dans une transaction sur Ethereum, permettait de faire en sorte qu'elle soit traitée plus vite. Mais ici, ce n'est pas possible étant donné qu'il n'y a pas de frais. C'est pour ça que EOSIO a mis en place un autre système qui est l'usage des ressources selon le nombre de EOS possédés par un utilisateur.

La plateforme compte trois types de ressources : la RAM, la bande passante du réseau et la vitesse du CPU. Parmi ces trois ressources, seule la RAM doit être achetée avec les EOS, les deux autres sont calculées selon le nombre de EOS dans le porte-monnaie virtuel. Un utilisateur aura alors une vitesse CPU et de la bande passante réseau déterminée par le nombre de EOS qu'il possède. Il devra acheter à part de la RAM s'il veut en avoir plus que ce qui est proposé de base. Lorsqu'une transaction sera faite, ces trois informations vont être transmises et les BP vont alors traiter les transactions selon les ressources de l'utilisateur qui l'envoie. Ce qui veut dire que plus on investit dans la blockchain, plus on possède d'EOS, plus nos transactions seront traitées rapidement.

Ce système s'avère être très utile pour les DApps qui n'ont plus besoin de payer à chacune de leurs transactions et faire payer leur client. Le seul problème est que le nombre de ressources selon les EOS peut varier avec le cours de la cryptomonnaie. Par exemple, 5 EOS peuvent valoir à un instant une vitesse CPU de 20hz et quelques jours plus tard seulement 5hz.

EOSIO a aussi d'autres particularités comme le fait de pouvoir upgrader les DApps qui sont déployées sur la blockchain. Les smart contracts ont également plus de fonctionnalités permettant par exemple de créer un système de permissions sur ceux-ci pour que seules certaines personnes puissent y accéder.

Grâce à ces systèmes, EOSIO permet de mettre en place des DApps, tout comme Ethereum. Ses points fort étant un plus grand nombre de transactions par seconde grâce notamment à la méthode de DPOS et des frais de transactions qui sont inexistants.

3.3 Hyperledger Fabric

Hyperledger est un logiciel open source permettant de créer des plateformes de blockchain. Il a été créé par une association d'institution tel qu'IBM, Intel ou encore Cisco, et a été initié par la Fondation Linux en 2015. Hyperledger Fabric est donc une plateforme de blockchain développée grâce au logiciel Hyperledger conçu pour être utilisé par des entreprises. Plus de 35 entreprises et presque 200 développeurs

participent activement à son maintien et développement. Fabric permet aux entreprises de créer leur propre blockchain, il n'y a pas comme sur Ethereum ou EOSIO une « main » blockchain que tout le monde utilise.

Figure 13 : Applications Hyperledger

(hyperledger.org, 2022)

Une des particularités de la plateforme Fabric est qu'elle a une architecture hautement modulable permettant ainsi la création d'applications optimisées pour des entreprises dans divers domaine tel que la finance, la santé, les ressources humaines ou encore les assurances. Fabric, étant fait pour des usages en entreprise, est donc une blockchain permissioned et privé. Ce qui veut dire qu'il n'y aura dans le réseau que des membres connus et vérifiés.

> *« This means that while the participants may not* fully *trust one another (they may, for example, be competitors in the same industry), a network can be operated under a governance model that is built off of what trust* does *exist between participants, such as a legal agreement or framework for handling disputes. »*
>
> *(Hyperledger Fabric Documentation)*

Ayant une architecture hautement modulable, il est également possible de ne pas mettre un protocole de consensus (proof of work, proof of stake, etc.) sur la blockchain. Ceci est possible car, étant permissioned, Fabric n'accueille que des membres de confiance. Ceci permettrait alors d'avoir une performance accrue. Étant permissioned, Fabric ne nécessite pas, contrairement à Ethereum et EOSIO qui sont publiques, de cryptomonnaie. La cryptomonnaie étant là principalement pour récompenser les mineurs qui valident les transactions et crée les blocs, il n'est pas nécessaire d'avoir ce système dans une blockchain permissioned ou ce travail sera effectué quoi qu'il arrive.

Fabric a également un système différent pour les transactions. Dans une blockchain « classique » comme Ethereum ou Bitcoin, les transactions se font selon l'architecture « order-execute ». Cette architecture va faire qu'une transaction va d'abord être ajoutée à la blockchain puis être distribuée à tous les nœuds du réseau. Une fois cela fait, la transaction sera exécutée. C'est pour cela que dans un système utilisant le protocole de consensus comme proof of work, une transaction prend beaucoup de temps. Proof of stake réduit largement ce temps. Fabric amène une architecture différente à ce problème : le « execute-order-validate ». Dans cette architecture, les nœuds qui valident une transaction vont d'abord exécuter la transaction puis se concerter entre eux (consensus) pour savoir s'ils ont le même résultat d'exécution. Si assez de validators ont le même résultat, alors la transaction sera ajoutée à la blockchain et distribuée au reste du réseau. Une architecture pareille permet l'exécution parallèle de plusieurs transactions et ainsi augmente la vitesse des TPS.

Comme dit plus tôt, Fabric à une architecture modulable, ce qui veut dire que les fonctionnalités sont des modules qui peuvent être ajoutés au code de la blockchain selon les besoins. Fabric possède notamment les modules suivants :

- Système pouvant accueillir plusieurs BDD connues comme SQL, SQLite, etc.
- Protocole de consensus.
- Système d'identification.
- Système de smart contracts, appelé chaincode.
- Système de « tunnel » accessible à certains nœuds et invisible à d'autres.

Plusieurs autres modules sont disponibles, mais ceux-ci sont les principaux. Tous ces systèmes font d'Hyperledger Fabric une des plateformes de blockchain permissioned les plus performantes. Fabric peut atteindre jusqu'à 20'000 TPS.

3.4 Comparaison

Il existe bien évidemment plusieurs autres plateformes de blockchain ayant chacune leur spécificité. Les trois que nous avons vu, représentent chacune une approche différente au problème de performance, sécurité et fiabilité. Voici un tableau qui résume les différences entre chaque plateforme.

Tableau 1 : Comparaison plateformes Blockchain

	Ethereum	EOSIO	Fabric
Permissioned / Permissionless	Permissionless	Permissionless	Permissioned
Protocole consensus	Proof of stake	Delegated proof of stake	Au choix
Vitesse TPS	100 - 200	10’000	20’000
DApps possible	OUI	OUI	OUI
Cryptomonnaie	OUI	OUI	Possible mais pas nécessaire
Modification DApps	NON	OUI	?
Langage de programmation spécifique	OUI	OUI	NON
Frais de transaction	OUI	NON	NON

4. Méthode concrète

Dans ce chapitre, nous allons utiliser Hyperledger Fabric pour mettre en place un environnement de test qui permettra de comprendre les concepts de base de Fabric. Pour ce faire, nous allons utiliser le « test network » qui est proposé directement par Fabric pour en faire une utilisation personnalisée.

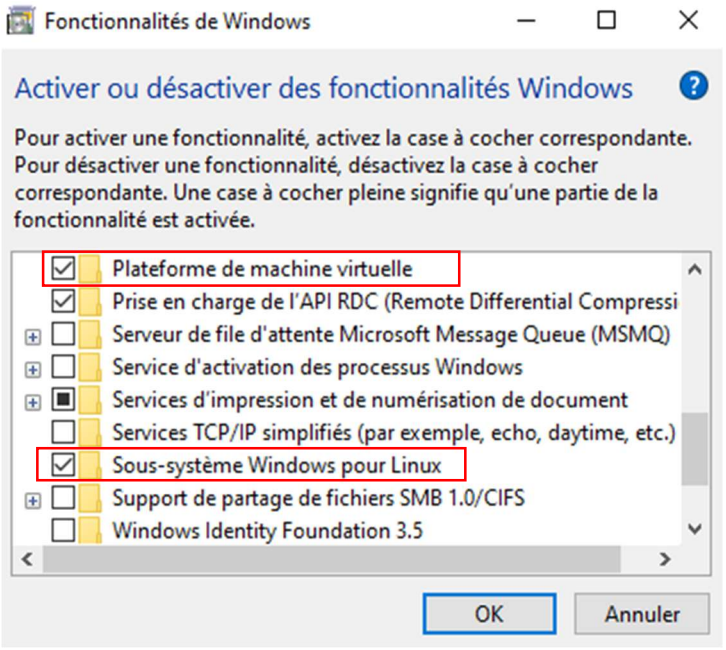
Il faut avant toute chose savoir qu'Hyperledger Fabric n'est déployable que dans un environnement Linux. La plupart des ordinateurs étant sous Windows, nous allons donc devoir mettre en place une machine virtuelle (VM) Linux Ubuntu fonctionnant sur Windows grâce à une fonctionnalité appelé WSL.

4.1 Prérequis

Nous allons commencer par installer tous les prérequis nécessaires pour la suite. Il faut tout d'abord vérifier si notre version Windows à la fonctionnalité WSL. Pour ce faire, il faut appuyer sur les touches Windows + R et écrire winver. Une fenêtre apparaîtra alors avec les informations sur la version. La fonctionnalité WSL n'est disponible qu'à partir des versions Windows 10 19'041 ou plus. Si la version de votre OS est inférieure, il faudra alors passer par logiciel comme VirtualBox ou VMware.

Ensuite, nous devons regarder si la fonctionnalité WSL est activée sur notre OS. Pour faire cela, il faut écrire dans la barre de recherche en bas à gauche « fonctionnalités de Windows ». Nous arriverons alors sur cette fenêtre :

Figure 14 : Fonctionnalités Windows à cocher



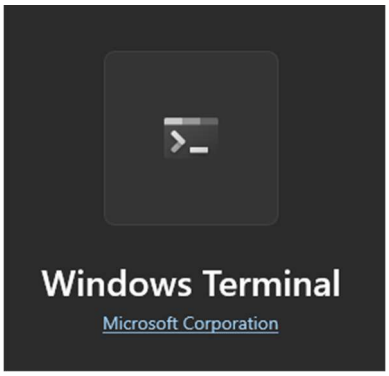
Il faut alors vérifier que les cases « Plateforme de machine virtuelle » et « Sous-système Windows pour linux » soient bien cochées. Une fois cela fait, nous devons ouvrir une ligne de commande PowerShell et copier la ligne suivante pour installer WSL ainsi qu'Ubuntu :

```
$ wsl –install -d ubuntu
```

Après l'installation, un terminal de commande va se lancer pour demander un nom d'utilisateur et un mot de passe pour la VM Linux.

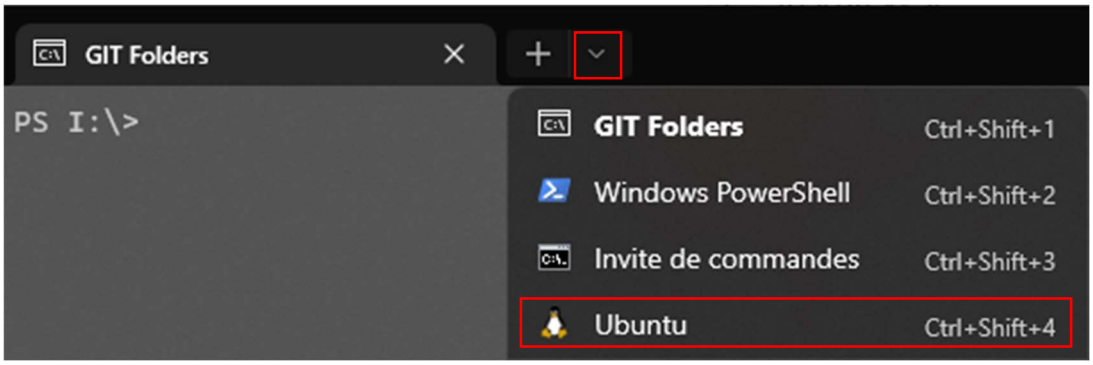
Il est conseillé d'installer Windows terminal qui a un meilleur aspect graphique qu'un terminal classique et qui permet certaines configurations :

Figure 15 : Windows Terminal



Nous pouvons désormais ouvrir le terminal Ubuntu avec Windows terminal en cliquant sur la flèche qui descend et sélectionner Ubuntu

Figure 16 : Ubuntu sur Windows Terminal

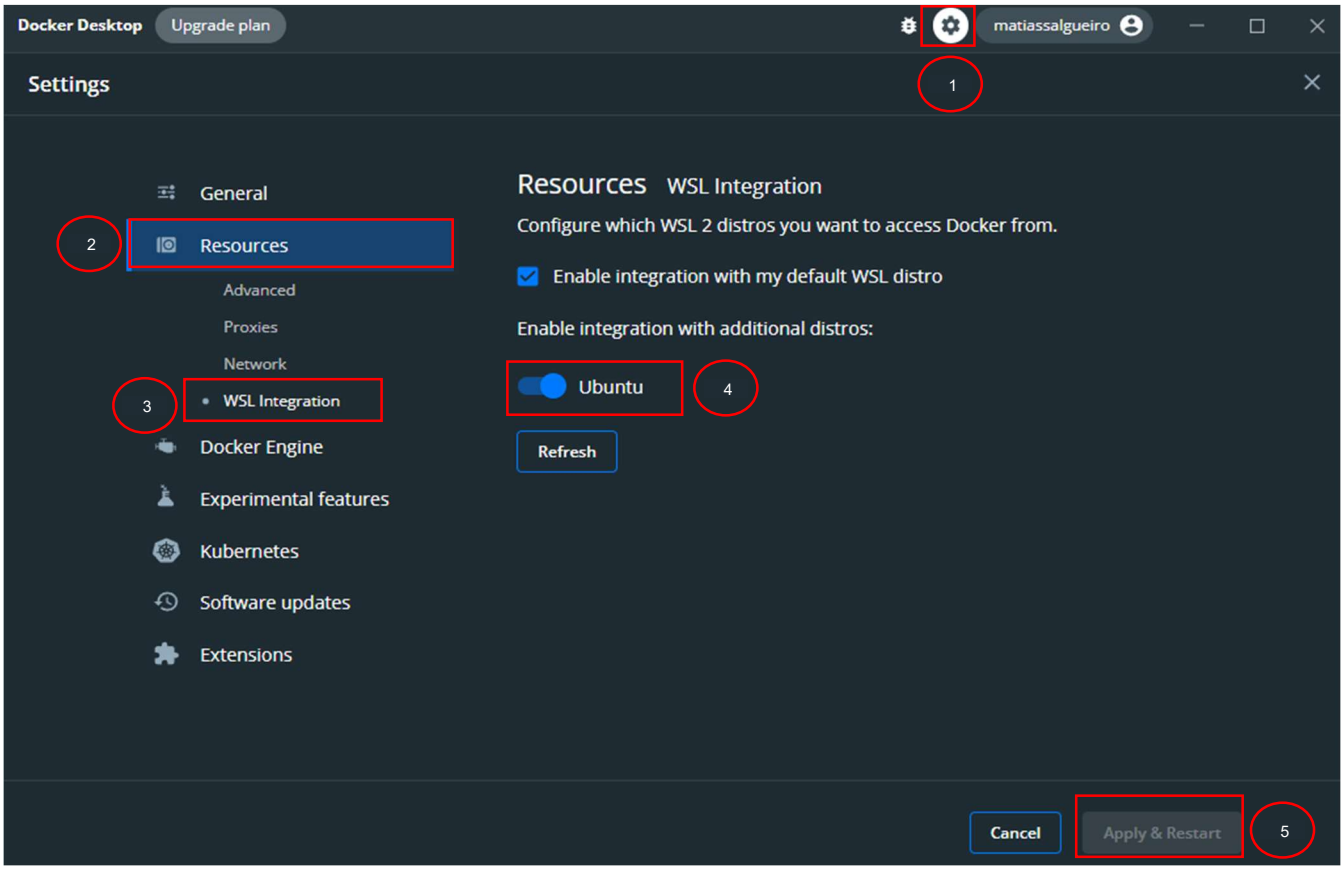


S'il n'y a pas l'option Ubuntu, il faut alors aller dans les paramètres – profils – Ubuntu et décocher l'option « masquer le profil de la liste déroulante ». Il est également possible dans l'onglet démarrage de mettre le profil Ubuntu par défaut pour que celui-ci soit ouvert lors du démarrage de Windows terminal. Pour le bon déroulement de cette méthode, il est préférable de modifier le paramètre « répertoire de démarrage », dans l'onglet

Ubuntu, et le mettre dans un emplacement connu. Celui utilisé pour cette méthode sera le bureau Windows « C:\Users\user\Desktop ».

Ensuite, il faut installer le logiciel Docker Desktop[6]. Pour que Docker puisse fonctionner avec Ubuntu, il faut aller changer les paramètres selon la figure ci-dessous :

Figure 17 : Paramètres Docker pour intégration à Ubuntu



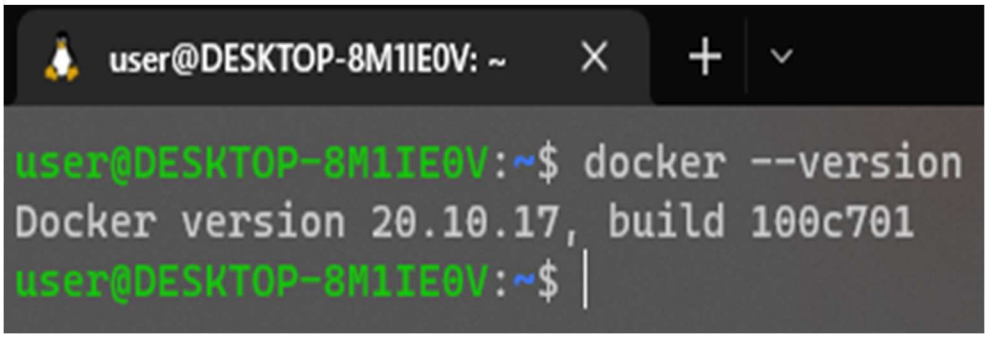
Après toutes ces étapes, un redémarrage de Windows est nécessaire. Si nous allons maintenant sur notre VM Ubuntu et entrons la ligne suivante :

```
$ docker --version
```

[6] DOCKER, 2022. Install Docker Desktop on Windows. *Docs.docer.com* [en ligne]. [Consulté le 5 septembre 2022]. Disponible à l'adresse suivante : Install Docker Desktop on Windows | Docker Documentation

Nous devrions obtenir ce résultat :

Figure 18 : Docker version



L'étape suivante est l'installation de Git, cURL et nodeJs sur notre VM :

```
$ sudo apt update
$ sudo apt-get update
$ sudo apt-get install git
$ sudo apt-get install curl
$ sudo apt install nodejs npm
```

Les deux premières lignes permettent de mettre à jour les packages lists. Enfin, il est également conseillé d'installer un éditeur de code comme VS Studio Code qui permettra de modifier plus facilement les fichiers javascript que nous allons utiliser.

4.2 Installation Fabric et Fabric Samples

Cette étape va installer les composants pour le fonctionnement de Fabric 2.2.4 ainsi que les exemples qui nous serviront à mettre en place l'environnement de test :

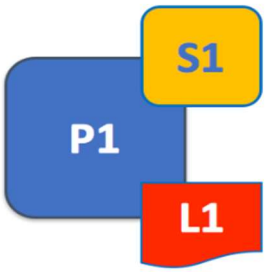
```
$ mkdir testenvfabric
$ cd testenvfabric
$ curl -sSLO
https://raw.githubusercontent.com/hyperledger/fabric/main/scripts/install-
fabric.sh && chmod +x install-fabric.sh
$ ./install-fabric.sh --fabric-version 2.2.4
```

Ces lignes nous permettent de créer un répertoire, s'y déplacer, copier le fichier d'installation de Fabric et installer la version 2.2.4.

4.3 Explication des concepts de Hyperledger Fabric

Fabric se compose de plusieurs éléments qu'il est important de comprendre pour pouvoir utiliser Fabric plus facilement. Premièrement, il y a ce qu'on appelle les « Peer Nodes » qui sont tout simplement les nœuds du réseau. Chaque nœud peut contenir, une copie d'un registre de données (Ledger) ou bien plusieurs copies de différents registres. Un nœud peut également contenir un smart contract qui se nomme sur Fabric « Chaincode ». Ce sont les smart contract qui permettent l'interaction avec le registre de données. Voici un exemple schématique qui montre à quoi ressemblerait un nœud. P pour Peer Node, L pour Ledger et S pour Smart contract.

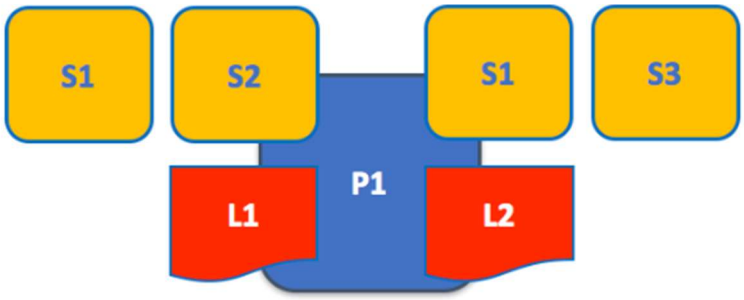
Figure 19 : Hyperledger Fabric Peer Node, Smart Contract et Ledger



(Hyperledger Fabric documentation – Peers, 2022)

Deuxième schéma cette fois-ci avec deux Ledger et quatre Smart contract sur un Node :

Figure 20 : Fabric Node avec multiples Smart contract et Ledger

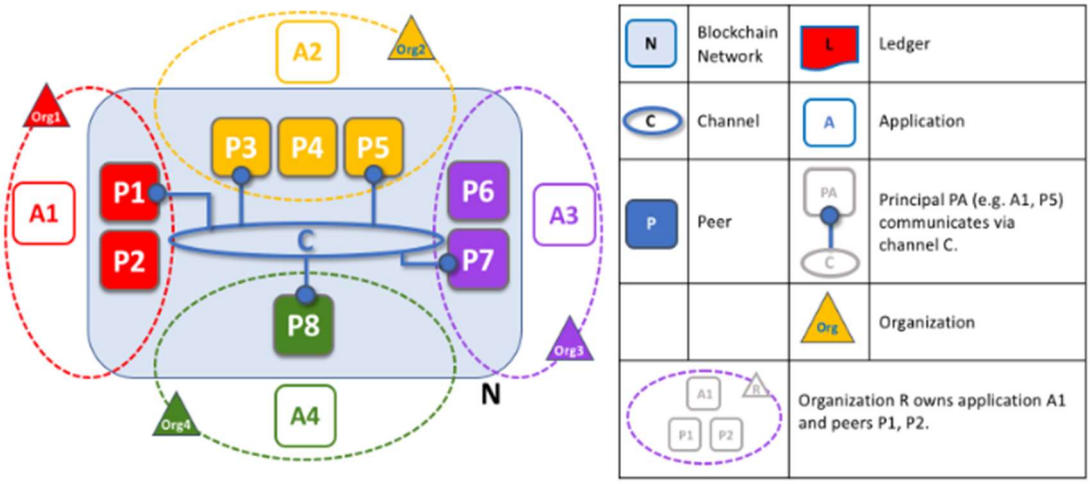


(Hyperledger Fabric documentation – Peers, 2022)

Il y a ensuite le concept de tunnel entre les nœuds qu'on appelle « Channel » et qui permet de relier plusieurs nœuds les uns aux autres. Ce Channel peut autoriser l'accès à un membre et la refuser à un autre pour ainsi pouvoir communiquer en privé. Un nœud peut être relié à plusieurs Channel en même temps. Pour accéder au contenu d'un nœud, il faut passer par une application cliente. Cela peut être une page web, application Windows, etc. Fabric met à disposition un « Fabric Gateway service » qui sert d'API pour pouvoir accéder aux nœuds d'un réseau. Généralement, les nœuds sont supervisés par

ce qu'on appelle des organisations. Ces organisations peuvent être une entreprise, un magasin, etc. qui sert d'entité mère aux nœuds. Voici maintenant un schéma qui représenterait un réseau entre quatre organisations :

Figure 21 : Schéma Fabric avec quatre organisations différentes



(Hyperledger Fabric documentation – Peers, 2022)

Fabric utilise aussi ce qu'on appelle des « Orderer » qui vont recevoir les transactions des nœuds. L'Orderer va trier et ordonner les transactions et les mettre dans un block avant de les renvoyer aux nœuds pour qu'il soit ajouté à la blockchain.

Enfin, Fabric peut également utiliser un système d'identité qui se nomme « Certificate Authorities » ou CA, qui permet d'identifier les nœuds selon l'organisation à laquelle ils appartiennent grâce au système de clé publique et clé privée. L'utilité du CA est donc d'identifier les nœuds, mais également de pouvoir vérifier si un nœud a le droit de faire une action, comme valider une transaction, ou pas.

La différence entre CA et Channel est qu'un Channel est simplement un tunnel connectant plusieurs nœuds entre eux, mais sans vérifier ce qu'il se passe dedans. Les CA permettent, eux, de contrôler qui a le droit de faire une action ou pas. Les CA couplés au Channel permettent donc une sécurité renforcée. Ce système n'est pas indispensable pour le bon fonctionnement d'un réseau.

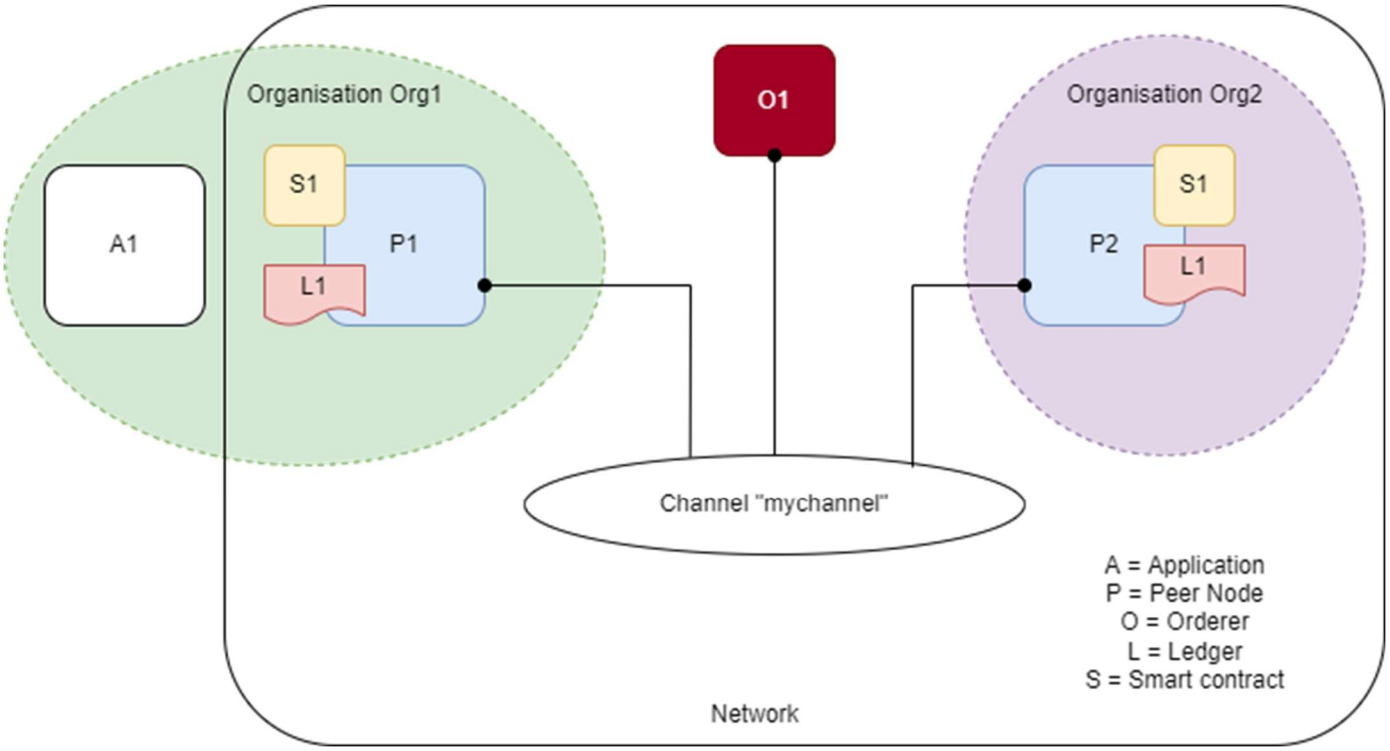
Pour résumer : un nœud peut contenir un ou plusieurs smart contract appelés Chaincode. Il peut également contenir une copie d'un ou plusieurs registres différents. Les nœuds peuvent être reliés entre eux par des Channels permettant ainsi de ne pas partager un registre ou un smart contract avec tous les nœuds d'un réseau. Les nœuds qui sont reliés par des channels auront alors chacun une copie du même registre ainsi que les mêmes smart contract. Pour accéder aux nœuds, il faut utiliser le Fabric

Gateway Service (FGS) qui agit comme une API. Lorsqu'un client veut accéder au contenu du registre d'un nœud ou bien utiliser un smart contract sur celui-ci, il va envoyer une requête à travers le FGS. La requête, ou transaction, sera transmise au nœud qui va l'exécuter puis la transmettre à un Orderer. Cet Orderer va ensuite regrouper plusieurs transactions, les valider et ordonner puis les mettre dans un block. Ce block sera ensuite renvoyé aux nœuds qui vont l'ajouter à leur registre.

4.4 Mise en place d'un environnement de test

Nous allons maintenant voir comment mettre en place le « test network » proposé par Fabric et comment l'utiliser. Cet environnement de test contient deux nœuds Peer Node appartenant chacun à une organisation (Org1 & Org2) ainsi qu'un Orderer. Les deux organisations et l'Orderer ont chacun un CA permettant de les identifier. Tous les composants de ce réseau sont déployés dans des conteneurs Docker Compose. Nous mettrons également en place un Channel appelé « mychannel ». Fabric permet de coder ses smart contracts en divers langages de programmation comme Go, Java, Javascript ou bien Type Script. Pour cette démarche, nous utiliserons Javascript. Finalement, nous verrons comment utiliser la blockchain grâce à une application en Javascript. Les organisations représenteront des librairies ayant un réseau partagé des livres en stock. Voici un schéma représentant le réseau test que nous allons mettre en place :

Figure 22 : Schéma environnement de test



Avant de commencer, il est important de rappeler que Hyperledger Fabric et hautement customisable. Le fichier configtx.yaml se trouvant dans le dossier /test-network/configtx permet de modifier de nombreux paramètres du réseau. Pour ne pas trop complexifier cette méthode, aucun paramètre de ce fichier ne sera modifié.

4.4.1 Lancement du réseau

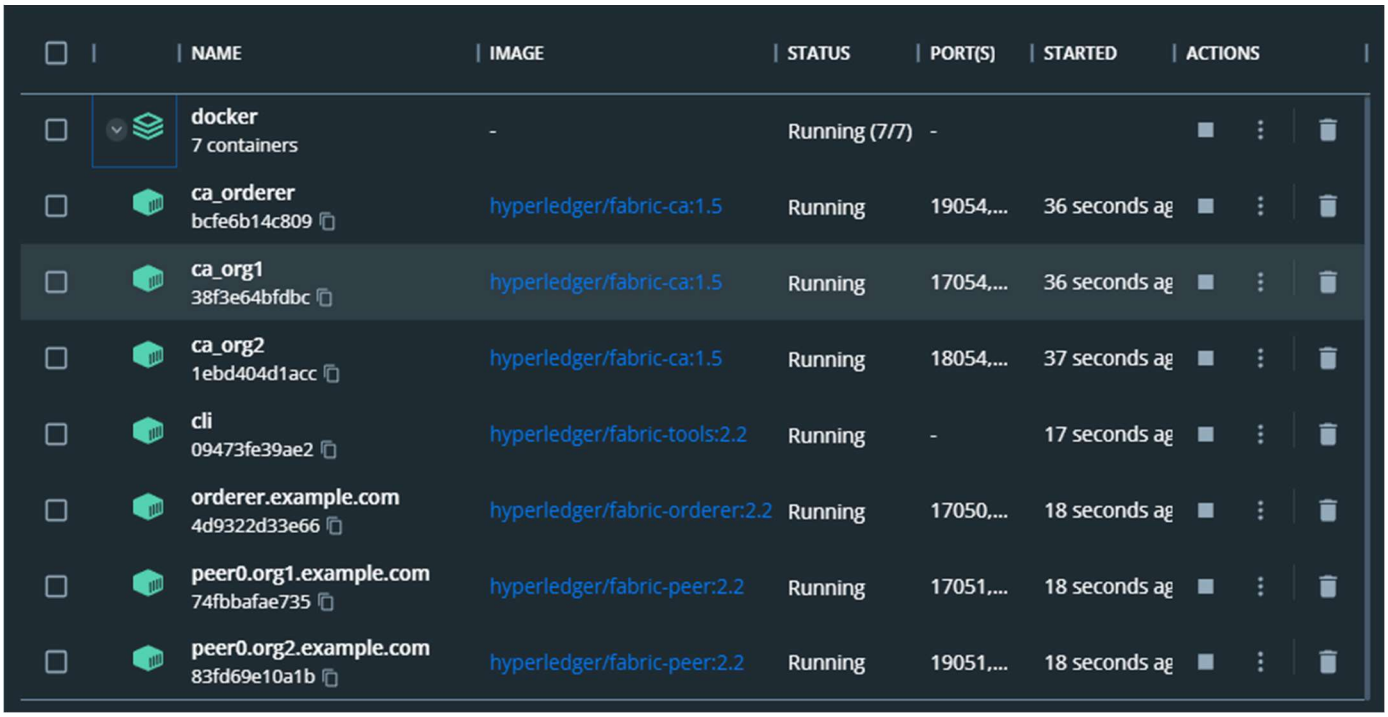
Pour commencer, il faut avoir lancé Docker Desktop et la VM Ubuntu. Une fois la VM lancée et son terminal ouvert, il faut aller dans le répertoire où nous avons installé l'environnement de test et démarrer le réseau :

```
$ cd fabric-samples/test-network
$ ./network.sh up -ca
```

Nous pouvons voir sur l'image ci-dessous que Docker a créé sept conteneurs :

- Un conteneur pour le CA de l'Orderer
- Un conteneur pour le CA de l'organisation Org1 et un autre pour le CA de l'organisation Org2
- Un conteneur qui simule un client que nous n'allons pas utiliser
- Un conteneur pour l'Orderer
- Un conteneur pour l'organisation Org1 et un autre pour l'Organisation Org2

Figure 23 : Docker avec les conteneurs du network



Nous allons ensuite créer le Channel pour que les nœuds Org1 et Org2 puissent communiquer.

```
$ ./network.sh createChannel
```

Cette commande permet de créer un Channel entre Org1 et Org2. Sans spécification quelconque, le nom du Channel sera « mychannel ».

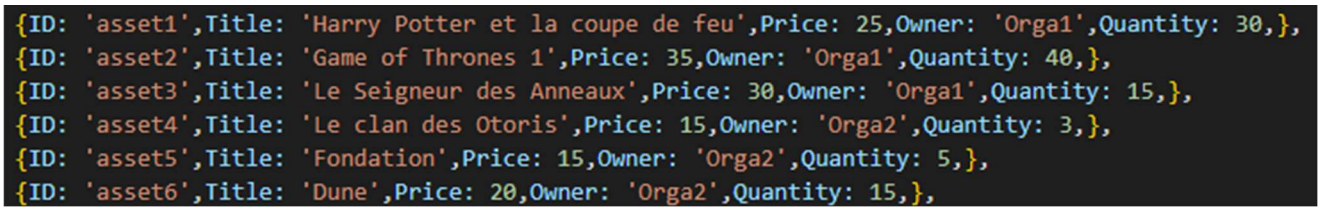
Le message suivant devrait apparaître si le Channel a bien été créé :

Figure 24 : Channel créer avec succès

4.4.2 Déploiement smart contract

L'étape suivante est de déployer un smart contract sur notre Channel. Nous allons donc utiliser un des exemples que Fabric nous met à disposition et qui se situe dans le dossier /fabric-samples/asset-transfer-basic/chaincode-javascript. Mais avant de l'utiliser, nous allons le modifier pour que celui-ci corresponde à notre Use-Case avec les librairies et les livres. Le code du smart contract se trouve à l'annexe 1.

Figure 25 : Données initial



Voici à quoi ressemble les données que nous allons utiliser. Dans Hyperledger Fabric, les objets sont appelés des « asset ». Un livre contient donc, son titre, son prix, quelle organisation le possède et enfin la quantité disponible. Les lignes suivantes sur la VM vont déployer le smart contract sur le réseau :

```
$ ./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-
javascript -ccl javascript
```

Figure 26 : Smart contract déployé avec succès

Le message ci-dessus indique que le Chaincode a bien été initialisé sur le Channel « mychannel ». Nous avons désormais tout qui est mis en place dans notre environnement de test. Il ne nous reste plus qu'à tester tout cela à l'aide d'une application.

4.4.3 Utilisation d'une application

Il y a dans le répertoire /fabric-samples/asset-transfer-basic un dossier se nommant /application-javascript. Nous allons modifier le app.js se trouvant dans ce dossier. Le code est à l'annexe 2 de ce document. Prenons un instant pour comprendre ce que le code va faire :

Figure 27 : Constantes Chaincode

```
const channelName = 'mychannel';
const chaincodeName = 'basic';
const mspOrg1 = 'Org1MSP';
const walletPath = path.join(__dirname, 'wallet');
const org1UserId = 'appUser';
```

Voici les paramètres de l'application. On peut voir qu'elle va être déployée sur le Channel « mychannel » et va utiliser le smart contract que nous avons modifier juste avant et qui se nomme « basic ». L'application va passer par le nœud de l'organisation Orga1.

Figure 28 : Connexion à la Gateway et création constante network et contract

```
const gateway = new Gateway();

try {
    await gateway.connect(ccp, {
        wallet,
        identity: org1UserId,
        discovery: { enabled: true, asLocalhost: true }
    });

    const network = await gateway.getNetwork(channelName);

    const contract = network.getContract(chaincodeName);
```

L'application va ensuite créer une instance de Gateway pour interagir avec le réseau de blockchain. On va ensuite récupérer le réseau sur lequel la Gateway est connectée ainsi que l'instance la plus importante : le contract. Ce contract va nous permettre d'accéder aux fonctions que nous avons coder dans le smart contract. Grâce aux fonctions du

smart contract, que nous utiliserons à travers l'application, nous allons pouvoir interagir avec les données du réseau.

4.4.3.1 Les étapes de l'application

Voici les étapes que l'application va faire lorsqu'elle sera lancée.

Figure 29 : Étapes application Javascript 1

```
1 console.log('\n--> Submit Transaction: InitLedger, function creates the initial set of assets on the ledger');
  await contract.submitTransaction('InitLedger');
  console.log('*** Result: committed');

2 console.log('\n--> Evaluate Transaction: GetAllAssets, function returns all the current assets on the ledger');
  let result = await contract.evaluateTransaction('GetAllAssets');
  console.log(`*** Result: ${prettyJSONString(result.toString())}`);

3 console.log('\n--> Submit Transaction: CreateAsset, creates new asset with ID, color, owner, size, and appraisedValue arguments');
  result = await contract.submitTransaction('CreateAsset', 'asset13', 'L\'homme Bicentenaire', '15', 'Orga2', '10');
  console.log('*** Result: committed');
  if (`${result}` !== '') {
      console.log(`*** Result: ${prettyJSONString(result.toString())}`);
  }
```

- Étape 1 : initialiser les données.
- Étape 2 : demander tous les assets disponibles pour les visualiser.
- Étape 3 : créer un nouvel asset.

Figure 30 : Étapes application Javascript 2

```
4 console.log('\n--> Evaluate Transaction: ReadAsset, function returns an asset with a given assetID');
  result = await contract.evaluateTransaction('ReadAsset', 'asset13');
  console.log(`*** Result: ${prettyJSONString(result.toString())}`);

  console.log('\n--> Evaluate Transaction: AssetExists, function returns "true" if an asset with given assetID exist');
5 result = await contract.evaluateTransaction('AssetExists', 'asset1');
  console.log(`*** Result: ${prettyJSONString(result.toString())}`);

6 console.log('\n--> Submit Transaction: UpdateAsset asset1, change the quantity from 30 to 20');
  await contract.submitTransaction('UpdateAsset', 'asset1', 'Harry Potter et la coupe de feu', 25, 'Orga1', 20);
  console.log('*** Result: committed');

7 console.log('\n--> Evaluate Transaction: ReadAsset, function returns "asset1" attributes');
  result = await contract.evaluateTransaction('ReadAsset', 'asset1');
  console.log(`*** Result: ${prettyJSONString(result.toString())}`);
```

- Étape 4 : demander l'asset nouvellement créé pour le visualiser.
- Étape 5 : tester si l'asset 1 existe.
- Étape 6 : update l'asset 1 en changeant la quantité de 30 à 20.
- Étape 7 : demander l'asset 1 pour voir les changements effectués.

Figure 31 : Étapes application Javascript 3

```
8   console.log('\n--> Submit Transaction: TransferAsset asset1, transfer to new owner of Tom');
    await contract.submitTransaction('TransferAsset', 'asset1', 'Orga2');
    console.log('*** Result: committed');

    console.log('\n--> Evaluate Transaction: ReadAsset, function returns "asset1" attributes');
9   result = await contract.evaluateTransaction('ReadAsset', 'asset1');
    console.log(`*** Result: ${prettyJSONString(result.toString())}`);
} finally {
10  gateway.disconnect();
    console.log('Application terminated');
```

- Étape 8 : changer le détenteur de l'asset 1, passer de Orga1 à Orga2.
- Étape 9 : demander l'asset 1 pour visualiser le changement.
- Étape 10 : déconnecter la Gateway et fermer l'application

Toutes ces étapes sont la simulation de ce que pourrait faire un utilisateur à travers une page web.

Voici maintenant les commandes Ubuntu pour lancer l'application lorsque l'on se trouve sur le répertoire /test-network :

```
$ cd ../asset-transfer-basic/application-javascript/
$ npm install
$ node app.js
```

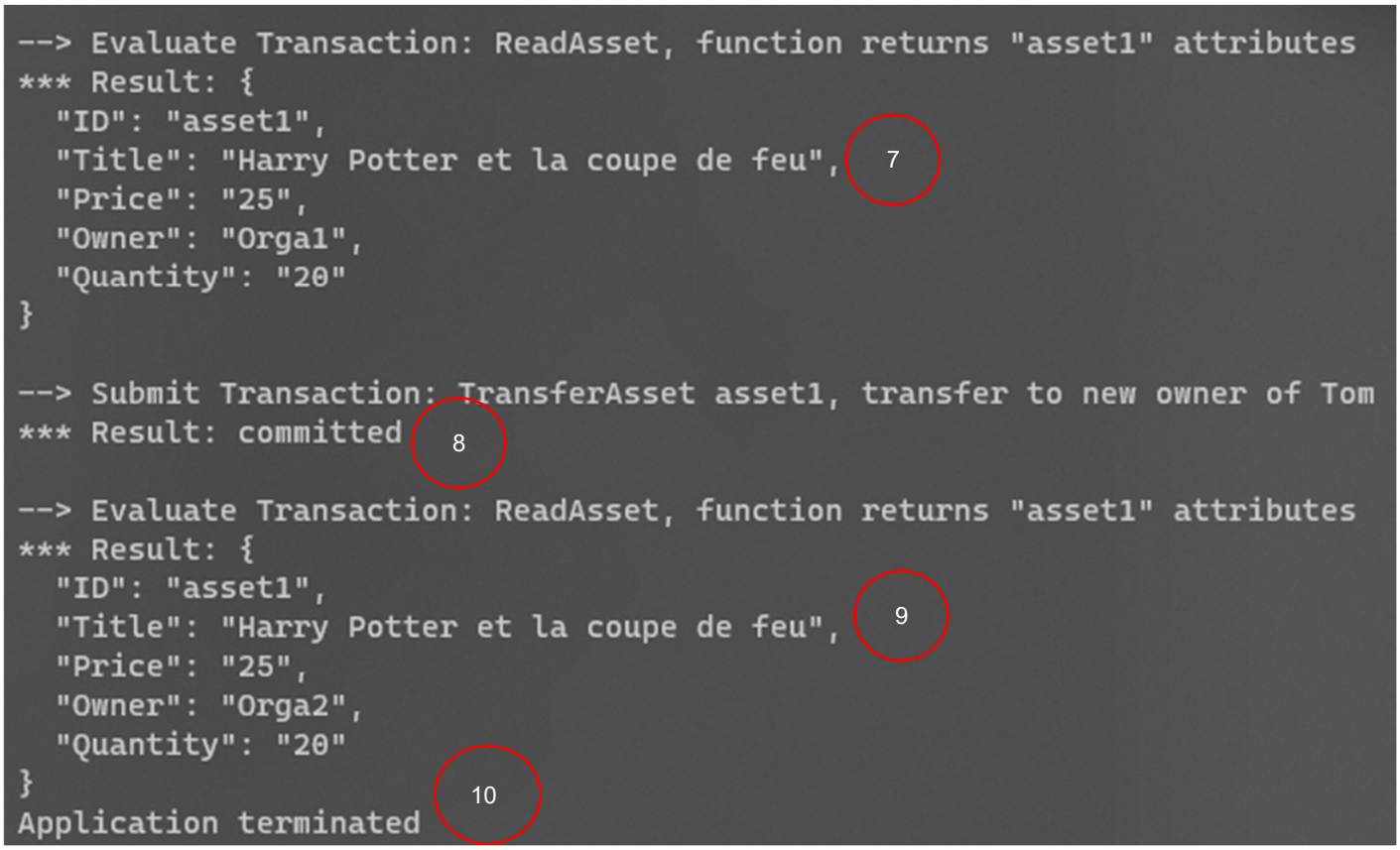
Les résultats qui devraient apparaitre sur le terminal sont les suivants (l'étape 1 et 2 se trouvent à l'annexe numéro 3) :

Figure 32 : Résultats étapes 3 à 6

```
--> Submit Transaction: CreateAsset, creates new asset with ID, color, owner, size, and appraisedValue arguments
*** Result: committed
*** Result: {
  "ID": "asset13",
  "Title": "L'homme Bicentenaire",
  "Price": "15",
  "Owner": "Orga2",
  "Quantity": "10"
}

--> Evaluate Transaction: ReadAsset, function returns an asset with a given assetID
*** Result: {
  "ID": "asset13",
  "Title": "L'homme Bicentenaire",
  "Price": "15",
  "Owner": "Orga2",
  "Quantity": "10"
}

--> Evaluate Transaction: AssetExists, function returns "true" if an asset with given assetID exist
*** Result: true

--> Submit Transaction: UpdateAsset asset1, change the quantity from 30 to 20
*** Result: committed
```

3 4 5 6

Figure 33 : Résultats étapes 7 à 10



Nous pouvons constater que les résultats obtenus sont bien ceux que nous voulions avoir. Grâce à cette méthode concrète, nous avons vu comment mettre en place un réseau de blockchain basique. Mais Hyperledger Fabric a, comme dit précédemment, une grande customisation possible. Il est par exemple possible de mettre en place des « Private Data Collection » qui sont des collections de données que seul celui qui les met sur le réseau peut consulter. Il y a dans le répertoire /fabric-samples/config trois fichiers permettant de personnaliser beaucoup de points importants du réseau. Le système de la base de données, quel algorithme de tri utilise les Orderer, définir des profils de règles pour les Channel et bien d'autres encore. Nous n'avons vu ici que les points fondamentaux pour pouvoir avoir un réseau simple et fonctionnel.

5. Conclusion

Le sujet de la blockchain est très vaste et en constante évolution, il est difficile de résumer tous les aspects de cette technologie dans un travail de Bachelor tant ses usages sont nombreux et diversifiés. Beaucoup d'experts parlent énormément de la blockchain comme étant la prochaine révolution numérique. Certains prédisent même quand le prochain internet sera décentralisé.

Comme nous avons pu le voir, la technologie de la blockchain propose des solutions à de nombreux problème existant dans notre société toujours plus numérisée. Décentralisation, immuabilité, anonymat et bien d'autres encore, sont les points fort de la blockchain qui font défaut aux infrastructures de sauvegarde de données « classique ». C'est pour cela qu'autant d'entreprises, petites comme grosses, s'intéressent à cette technologie et les solutions qu'elle apporte.

Mais bien qu'elle ait beaucoup d'avantages, la technologie de blockchain n'est pas parfaite. Ces désavantages principaux sont la vitesse ainsi que l'énorme utilisation d'électricité et capacité de computation. Un autre point souvent oublié est que la blockchain n'est pas très adaptée aux données lourdes comme des documents. Nous avons pu voir quelques méthodes permettant de réduire ces problèmes, mais sans pour autant les effacer complètement. Pour les blockchains publiques en tout cas.

Là où la blockchain démontre tout son potentiel est dans les réseaux permissioned et privée comme avec Hyperledger Fabric. Grâce à ce genre de plateformes, il est possible d'implémenter plus ou moins facilement un réseau de blockchain pour des entreprises. Mais il faut rappeler que la blockchain n'est utile que lorsque plusieurs participants sont dans le réseau. Monter une infrastructure sur la blockchain que pour une seule entreprise se trouvant dans un seul bâtiment n'aurait pas de sens, car tout les avantages de la blockchain seraient perdus.

Je pense que la blockchain est un sujet passionnant qui est malheureusement trop souvent incompris, car ses concepts sont difficiles à appréhender. Lorsque l'on parle de blockchain à quelqu'un, cette personne aura forcément en tête le Bitcoin ou bien les NFT et pensera que tout cela n'est fait que dans un but spéculatif pour gagner de l'argent. Alors que les usages de cette technologie sont très grands et apportent une réelle solution de gestion des données.

Moi-même avant d'effectuer ce travail, j'avais une opinion mitigée sur la blockchain et ses applications. Grâce à ce travail, j'ai pu comprendre ce qu'est la blockchain et quels enjeux tournent autour de cette technologie. Je pense que beaucoup d'entreprises

pourraient bénéficier des apports de la blockchain. Si l'on choisit de mettre en place un réseau de blockchain pour son entreprise, Hyperledger Fabric est une très bonne plateforme, accessible et polyvalente.

Bibliographie

ARORA, Shivam, 2022. « What is Bitcoin mining : how does it work, proof of work, mining hardware and more. ». *Simplilearn* [en ligne]. 15 septembre 2022. [Consulté le 18 septembre 2022]. Disponible à l'adresse : What Is Bitcoin Mining: How Does It Work, Proof of Work and More | Simplilearn

BAILEY, David, 2022. « 19 Blockchain application use cases that will surprise you. ». *Supplain* [en ligne]. 30 avril 2022. [Consulté le 10 août 2022]. Disponible à l'adresse : 19 Blockchain application use cases that will surprise you (supplain.io)

BANK OF ENGLAND, 2018. How are cryptocurrencies created ? [enregistrement vidéo]. *YouTube [en ligne]* 3 octobre 2018. [Consulté le 1 septembre 2022]. Disponible à l'adresse : How are cryptocurrencies created? - YouTube

Bitcoin, *Wikipédia : l'encyclopédie libre* [en ligne]. Dernière modification de la page le 2 septembre 2022 à 09:48. [Consulté le 2 septembre 2022]. Disponible à l'adresse : https://en.wikipedia.org/w/index.php?title=Bitcoin&oldid=1108063744

BITPANDA, 2022. Pourquoi dois-je payer des frais de transaction ? *Bitpanda* [en ligne]. [Consulté le 10 août 2022]. Disponible à l'adresse : Pourquoi dois-je payer des frais de transaction ? — Bitpanda Academy

BITPANDA, 2022. Qu'est-ce qu'une DApp ? *Bitpanda* [en ligne]. [Consulté le 10 août 2022]. Disponible à l'adresse : Qu'est-ce qu'une DApp ? — Bitpanda Academy

BITPANDA, 2022. Qu'est-ce qu'un mining pool et quel est son fonctionnement ? *Bitpanda* [en ligne]. [Consulté le 10 août 2022]. Disponible à l'adresse : Qu'est-ce qu'un mining pool et quel est son fonctionnement ? — Bitpanda Academy

BLOCKGEEKS, 2019. What is EOS ? How Does it Work ? [enregistrement vidéo]. *YouTube [en ligne]* 8 février 2019. [Consulté le 10 septembre 2022]. Disponible à l'adresse : What is EOS? How Does it Work? - YouTube

Blockchain, *Wikipédia : l'encyclopédie libre* [en ligne]. Dernière modification de la page le 5 septembre 2022 à 00:45. [Consulté le 5 septembre 2022]. Disponible à l'adresse : https://en.wikipedia.org/w/index.php?title=Blockchain&oldid=1108541547

CHAUM, David, 1979. *Computer systems established, maintained, and trusted by mutually suspicious groups* [en ligne]. 22 février 1979. [Consulté le 15 août 2022]. Disponible à l'adresse : techrep.pdf (chaum.com)

CNBCTV18, 2022. « 5 Key events that must occur for Ethereum to hit 100K transactions per second. ». *CNBC TV 18* [en ligne]. 8 septembre 2022. [Consulté le 10 septembre 2022]. Disponible à l'adresse : 5 Key Events That Must Occur For Ethereum To Hit 100k Transactions Per Second (cnbctv18.com)

COINBASE, 2019. Coinbase Earn : What is Delegated Proof of Stake ? [enregistrement vidéo]. *YouTube [en ligne]* 3 juin 2019. [Consulté le 22 août 2022]. Disponible à l'adresse : Coinbase Earn: What is Delegated Proof of Stake? (Lesson 2 of 5) - YouTube

CONTI, Robyn, SCHMIDT, John, 2022. « What is an NFT ? Non-Fungible Tokens Explained. ». *Forbes* [en ligne]. 7 avril 2022. [Consulté le 10 août 2022]. Disponible à l'adresse : 6 Top Cryptocurrencies With Smart Contracts | Nasdaq

CRYPTOAST, 2019. Comprendre la blockchain en 7 minutes [enregistrement vidéo]. *YouTube [en ligne]* 8 mai 2019. [Consulté le 25 août 2022]. Disponible à l'adresse : COMPRENDRE LA BLOCKCHAIN EN 7 MINUTES - YouTube

Cryptocurrency, *Wikipédia : l'encyclopédie libre* [en ligne]. Dernière modification de la page le 2 septembre 2022 à 17:23. [Consulté le 2 septembre 2022]. Disponible à l'adresse : https://en.wikipedia.org/w/index.php?title=Cryptocurrency&oldid=1108123094

Cryptographic hash function, *Wikipédia : l'encyclopédie libre* [en ligne]. Dernière modification de la page le 1 septembre 2022 à 21:35. [Consulté le 2 septembre 2022]. Disponible à l'adresse : https://en.wikipedia.org/w/index.php?title=Cryptographic_hash_function&oldid=1107978246

DALY, Lyle, 2022. « How many Cryptocurrencies are there ? ». *The Motley Fool* [en ligne]. 27 juin 2022. [Consulté le 10 août 2022]. Disponible à l'adresse : How Many Cryptocurrencies Are There? (fool.com)

DARPA, 2022. Darpa-funded study provides insights into Blockchain vulnerabilities *Darpa* [en ligne]. 21 juin 2022. [Consulté le 10 août 2022]. Disponible à l'adresse : DARPA-Funded Study Provides Insights into Blockchain Vulnerabilities

David Chaum, *Wikipédia : l'encyclopédie libre* [en ligne]. Dernière modification de la page le 2 septembre 2022 à 17:52. [Consulté le 2 septembre 2022]. Disponible à l'adresse : https://en.wikipedia.org/w/index.php?title=David_Chaum&oldid=1108127384

DigiCash, *Wikipédia : l'encyclopédie libre* [en ligne]. Dernière modification de la page le 14 mars 2022 à 06:24. [Consulté le 10 aout 2022]. Disponible à l'adresse : https://en.wikipedia.org/w/index.php?title=DigiCash&oldid=1077031959

Digital signature, *Wikipédia : l'encyclopédie libre* [en ligne]. Dernière modification de la page le 26 aout 2022 à 18:21. [Consulté le 29 aout 2022]. Disponible à l'adresse :

https://en.wikipedia.org/w/index.php?title=Digital_signature&oldid=1106835040

DOCKER, 2022. Install Docker Desktop on Windows. *Docs.docer.com* [en ligne]. [Consulté le 5 septembre 2022]. Disponible à l'adresse suivante : Install Docker Desktop on Windows | Docker Documentation

DUTTA, Bhumika, 2022. « Advantages and Disadvantages of Ethereum. ». *Analytics Steps* [en ligne]. 24 mars 2022. [Consulté le 3 septembre 2022]. Disponible à l'adresse : Advantages and Disadvantages of Ethereum | Analytics Steps

EOSIO, 2022. FAQ EOSIO *Eos.io* [en ligne]. [Consulté le 28 août 2022]. Disponible à l'adresse : FAQ – EOSIO

ETHEREUM, 2022. Private Ethereum for entreprise *Ethereum.org* [en ligne]. 15 septembre 2022. [Consulté le 15 août 2022]. Disponible à l'adresse : Private Ethereum for Enterprise | ethereum.org

Ethereum, *Wikipédia : l'encyclopédie libre* [en ligne]. Dernière modification de la page le 20 septembre 2022 à 14:05. [Consulté le 20 septembre 2022]. Disponible à l'adresse : https://en.wikipedia.org/w/index.php?title=Ethereum&oldid=1111342113

FABRION, Maxence, 2022. « The Merge : Ethereum est officiellement entré dans une nouvelle ère. ». *Les numériques* [en ligne]. 15 septembre 2022. [Consulté le 15 septembre 2022]. Disponible à l'adresse : The Merge : Ethereum est officiellement entré dans une nouvelle ère - Les Numériques (lesnumeriques.com)

FRANKENFIELD, Jake, 2022. « Proof-of-Stake (POS) ». *Investopedia* [en ligne]. 9 juin 2022. [Consulté le 5 septembre 2022]. Disponible à l'adresse : Proof-of-Stake (PoS) Definition (investopedia.com)

FRANKENFIELD, Jake, 2022. « Initial Coin offering (ICO) : Coin launch defined, with examples. ». *Investopedia* [en ligne]. 18 août 2022. [Consulté le 8 septembre 2022]. Disponible à l'adresse : Initial Coin Offering (ICO): Coin Launch Defined, with Examples (investopedia.com)

GUIGNANT, Arnaud, 2021. Blockchain : la base des cryptomonnaies. *Gafish* [en ligne]. 1 septembre 2021. [Consulté le 10 août 2022]. Disponible à l'adresse : Blockchain : la base des cryptomonnaies - Arnaud Guignant (gafish.fr)

Hashcash, *Wikipédia : l'encyclopédie libre* [en ligne]. Dernière modification de la page le 2 mai 2022 à 23:21. [Consulté le 20 aout 2022]. Disponible à l'adresse : https://en.wikipedia.org/w/index.php?title=Hashcash&oldid=1085880736

HAYES, Adam, 2022. « Blockchain Facts : What is it, how it works, and how it can be used. ». *Investopedia* [en ligne]. 18 septembre 2022. [Consulté le 18 septembre 2022]. Disponible à l'adresse : Blockchain Facts: What Is It, How It Works, and How It Can Be Used (investopedia.com)

HERTIG, Alyssa, 2022. « What is proof-of-work ? ». *Coindesk* [en ligne]. 16 décembre 2020. [Consulté le 2 septembre 2022]. Disponible à l'adresse : What Is Proof-of-Work? - CoinDesk

HYPERLEDGER, 2022. Hyperledger – Open Source Blockchain Technologies *hyperledger.org* [en ligne]. [Consulté le 5 septembre 2022]. Disponible à l'adresse : Hyperledger – Open Source Blockchain Technologies

HYPERLEDGER FABRIC, 2022. Hyperledger Fabric main documentation. *Hyperledger-fabric.readthedocs.io* [en ligne]. [Consulté le 5 septembre 2022]. Disponible à l'adresse : A Blockchain Platform for the Enterprise — hyperledger-fabricdocs main documentation

IBM BLOCKCHAIN, 2017. IBM and Maersk demo : Cross-border supply chain solution on blockchain [enregistrement vidéo]. *YouTube [en ligne]* 15 mars 2017. [Consulté le 25 août 2022]. Disponible à l'adresse : IBM and Maersk demo: Cross-border supply chain solution on blockchain - YouTube

IBM, 2020. *IBM Blockchain services for supply chain* [ligne]. Octobre 2020. [Consulté le 15 août 2022]. Disponible à l'adresse : APGWOG5A (ibm.com)

IBM, 2022. Qu'est-ce que Hyperledger Fabric ? *IBM* [en ligne]. [Consulté le 10 août 2022]. Disponible à l'adresse : Qu'est -ce que Hyperledger Fabric ? | IBM

IBM, 2022. What is Blockchain ? *IBM* [en ligne]. [Consulté le 10 août 2022]. Disponible à l'adresse : Qu'est-ce que la technologie de blockchain ? - IBM Blockchain | IBM

IBM, 2022. What is Blockchain security ? *IBM* [en ligne]. [Consulté le 10 août 2022]. Disponible à l'adresse : What is Blockchain Security? | IBM

INSIDER INTELLIGENCE, 2022. « The growing list of applications and use cases of blockchain technology in business and life. ». *Insider Intelligence* [en ligne]. 15 avril 2022. [Consulté le 10 août 2022]. Disponible à l'adresse : Use cases of blockchain technology in business and life (insiderintelligence.com)

Kalyanicynixit, 2020. Applications of Blockchain Technology. *Medium* [en ligne]. 3 mars 2020. [Consulté le 25 août 2022]. Disponible à l'adresse : Applications of Blockchain Technology | by Kalyanicynixit | Medium

KASPERSKY, 2022. What is Cryptocurrency and how does it work ? *Kaspersky* [en ligne]. 9 février 2022. [Consulté le 8 août 2022]. Disponible à l'adresse : What is cryptocurrency and how does it work? (kaspersky.com)

KAUR, Rashmeet, 2021. Which programming language is used in Ethereum Blockchain ? *Medium Data Driven Investor* [en ligne]. 8 juin 2021. [Consulté le 25 août 2022]. Disponible à l'adresse : Which programming language is used in Ethereum Blockchain? | by Rashmeet Kaur | DataDrivenInvestor

KRIPTOMAT, 2022. Une brève histoire de la technologie blockchain que tout le monde devrait lire. *Kriptomat* [en ligne]. 17 mars 2022. [Consulté le 10 août 2022]. Disponible à l'adresse : Une brève histoire de la technologie blockchain que tout le monde devrait lire (kriptomat.io)

LEEWAYHERTZ, 2019. What is EOS ? The complete guide to understanding EOS Blockchain. LeewayHertz [en ligne]. 2 juillet 2029. [Consulté le 28 août 2022]. Disponible à l'adresse : The Complete Guide to understanding EOS Blockchain | LeewayHertz

LISK, 2019. What is a Peer to Peer network ? Blockchain P2P Networks Explained [enregistrement vidéo]. *YouTube [en ligne]* 15 janvier 2019. [Consulté le 25 août 2022]. Disponible à l'adresse : What is a Peer to Peer Network? Blockchain P2P Networks Explained - YouTube

MCSHANE, Griffin, 2022. « What is a 51% attack ? ». *Coindesk* [en ligne]. 12 octobre 2021. [Consulté le 2 septembre 2022]. Disponible à l'adresse : What Is a 51% Attack? (coindesk.com)

NEWBERY, Emma, 2021. « 6 Top Cryptocurrencies with smart contracts ? ». *Nasdaq* [en ligne]. 21 septembre 2021. [Consulté le 10 août 2022]. Disponible à l'adresse : 6 Top Cryptocurrencies With Smart Contracts | Nasdaq

Non-fungible token, *Wikipédia : l'encyclopédie libre* [en ligne]. Dernière modification de la page le 19 septembre 2022 à 12:51. [Consulté le 19 septembre 2022]. Disponible à l'adresse : https://en.wikipedia.org/w/index.php?title=Non-fungible_token&oldid=1111129013

PARMAR, Dhruv, 2022. « 8 meilleures plateformes de blockchain pour créer des applications financières modernes. ». *Geekflare* [en ligne]. 31 mai 2022. [Consulté le 18 août 2022]. Disponible à l'adresse : 8 meilleures plateformes de blockchain pour créer des applications financières modernes (geekflare.com)

Peer-to-peer, *Wikipédia : l'encyclopédie libre* [en ligne]. Dernière modification de la page le 27 aout 2022 à 20:21. [Consulté le 29 aout 2022]. Disponible à l'adresse : https://en.wikipedia.org/w/index.php?title=Peer-to-peer&oldid=1107039482

POLLOCK, Darryn, 2018. « Which countries are best to start blockchain projects ? ». *Cointelegraph* [en ligne]. 9 août 2018 [Consulté le 15 août 2022]. Disponible à l'adresse : Which Countries Are Best to Start Blockchain Projects? (cointelegraph.com)

Proof of stake, *Wikipédia : l'encyclopédie libre* [en ligne]. Dernière modification de la page le 16 septembre 2022 à 13:02. [Consulté le 16 septembre 2022]. Disponible à l'adresse : https://en.wikipedia.org/w/index.php?title=Proof_of_stake&oldid=1110607293

Proof of work, *Wikipédia : l'encyclopédie libre* [en ligne]. Dernière modification de la page le 2 septembre 2022 à 17:10. [Consulté le 3 septembre 2022]. Disponible à l'adresse : https://en.wikipedia.org/w/index.php?title=Proof_of_work&oldid=1108121217

Public-key cryptography, *Wikipédia : l'encyclopédie libre* [en ligne]. Dernière modification de la page le 30 aout 2022 à 08:42. [Consulté le 2 septembre 2022]. Disponible à l'adresse :https://en.wikipedia.org/w/index.php?title=Public-key_cryptography&oldid=1107497738

QUENETAIN, Stanislas, 2019. Qu'est-ce que la Blockchain ? Une explication simple ! *Blockchains expert* [en ligne]. 8 février 2019. [Consulté le 10 août 2022]. Disponible à l'adresse : Qu'est ce que la blockchain? Une explication simple (blockchains-expert.com)

SCHOU-ZIBELL, Lotte, PHAIR, Nigel, 2018. How secure is Blockchain ? *Blogs.adb* [en ligne]. 4 mai 2018. [Consulté le 10 août 2022]. Disponible à l'adresse : How Secure is Blockchain? (adb.org)

SEGOUIN, Vincent, 2018. XebiCon'18 – Utiliser Hyperledger Fabric pour la création d'une blockchain privée [enregistrement vidéo]. *YouTube [en ligne]* 3 décembre 2018. [Consulté le 5 septembre 2022]. Disponible à l'adresse : XebiCon'18 - Utiliser Hyperledger Fabric pour la création d'une blockchain privée - YouTube

SIGALOS, Mackenzie, 2021. « Bitcoin mining has totally recovered from Chinese ban. ». *CNBC* [en ligne]. 10 décembre 2021. [Consulté le 10 août 2022]. Disponible à l'adresse : Bitcoin network hashrate hits all-time high after China crypto ban (cnbc.com)

SIMPLY EXPLAINED, 2017. Comment fonctionne une blockchain – Expliqué simplement [enregistrement vidéo]. *YouTube [en ligne]* 13 novembre 2017. [Consulté le 25 aout 2022]. Disponible à l'adresse : Comment fonctionne une blockchain - Expliqué simplement - YouTube

SIMIPLY EXPLAINED, 2017. Smart contracts – Simply Explained [enregistrement vidéo]. *YouTube [en ligne]* 20 novembre 2017. [Consulté le 23 août 2022]. Disponible à l'adresse : Smart contracts - Simply Explained - YouTube

SIMIPLY EXPLAINED, 2018. Proof-of-Stake (vs proof-of-work) [enregistrement vidéo]. *YouTube [en ligne]* 21 mars 2018. [Consulté le 19 août 2022]. Disponible à l'adresse : Proof-of-Stake (vs proof-of-work) - YouTube

SIMPLY EXPLAINED, 2018. Blockchains : how can they be used ? (Use cases for Blockchains) [enregistrement vidéo]. *YouTube [en ligne]* 29 mai 2018. [Consulté le 25 août 2022]. Disponible à l'adresse : Blockchains: how can they be used? (Use cases for Blockchains) - YouTube

SIMIPLY EXPLAINED, 2021. NFT's Explained in 4 minutes ! [enregistrement vidéo]. *YouTube [en ligne]* 8 juin 2021. [Consulté le 15 août 2022]. Disponible à l'adresse : NFT's Explained in 4 minutes! - YouTube

Single point of failure, *Wikipédia : l'encyclopédie libre* [en ligne]. Dernière modification de la page le 3 juillet 2022 à 07:23. [Consulté le 2 septembre 2022]. Disponible à l'adresse : https://en.wikipedia.org/w/index.php?title=Single_point_of_failure&oldid=1096247701

Smart contract, *Wikipédia : l'encyclopédie libre* [en ligne]. Dernière modification de la page le 7 septembre 2022 à 21:45. [Consulté le 10 septembre 2022]. Disponible à l'adresse : https://en.wikipedia.org/w/index.php?title=Smart_contract&oldid=1109087177

SUN, Sunny, 2017. What is digital signature ? [enregistrement vidéo]. *YouTube [en ligne]* 13 juin 2017. [Consulté le 20 août 2022]. Disponible à l'adresse : Qu'est-ce que la signature numérique? - YouTube

TELUSKO, 2019. What is Hyperledger ? [enregistrement vidéo]. *YouTube [en ligne]* 23 mars 2019. [Consulté le 10 septembre 2022]. Disponible à l'adresse : What is Hyperledger? - YouTube

TELUSKO, 2019. What is Hyperledger Fabric ? [enregistrement vidéo]. *YouTube [en ligne]* 5 décembre 2019. [Consulté le 10 septembre 2022]. Disponible à l'adresse : What is Hyperledger Fabric? | Blockchain - YouTube

TOTI, Benson, 2022. « Qu'est-ce que l'EOS – Une monnaie virtuelle ou bien plus ? ». *Coin Journal* [en ligne]. 3 août 2022. [Consulté le 6 septembre 2022]. Disponible à l'adresse : Qu'est-ce que Eos (EOS) & Quels sont ses usages - CoinJournal

TRAILS OF BITS, 2022. *Are blockchains decentralized ? Unintended centralities in distributed ledgers.* [en ligne]. Juin 2022. [Consulté le 15 août 2022]. Disponible à l'adresse : Are Blockchains Decentralized - Unintended Centralities in Distributed Ledgers - Final Report (website-files.com)

UBUNTU, 2022. Install Ubuntu on WSL2 on Windows 10. *Ubuntu.com* [en ligne]. [Consulté le 5 septembre 2022]. Disponible à l'adresse : Install Ubuntu on WSL2 on Windows 10 | Ubuntu

Visa, *Wikipédia : l'encyclopédie libre* [en ligne]. Dernière modification de la page le 13 septembre 2022 à 13:02. [Consulté le 13 septembre 2022]. Disponible à l'adresse : https://en.wikipedia.org/w/index.php?title=Visa_Inc.&oldid=1110078735

WOOD, Aaron, 2018. « West Virginia Secretary of States reports successful Blockchain voting in 2018 Midterm Elections. ». *Cointelegraph* [en ligne]. 17 novembre 2018. [Consulté le 10 août 2022]. Disponible à l'adresse : West Virginia Secretary of State Reports Successful Blockchain Voting in 2018 Midterm Elections (cointelegraph.com)

ZOLA, Andrew, 2021. Definition Hashing ? *Tech Target* [en ligne]. juin 2021. [Consulté le 25 août 2022]. Disponible à l'adresse : What is hashing and how does it work? (techtarget.com)

99BITCOINS, 2018. What is Ethereum ? A Beginner's Explanation in Plain English [enregistrement vidéo]. *YouTube [en ligne]* 26 juin 2018. [Consulté le 5 septembre 2022]. Disponible à l'adresse : What is Ethereum? A Beginner's Explanation in Plain English - YouTube

99BITCOINS, 2018. Ethereum Wallets Explained Simply (Smart Contracts, Gas, Transactions) [enregistrement vidéo]. *YouTube [en ligne]* 2 août 2018. [Consulté le 5 septembre 2022]. Disponible à l'adresse : Ethereum Wallets Explained Simply (Smart Contracts, Gas, Transactions) - YouTube

101 BLOCKCHAINS, 2020. « Permissioned Vs Permissionless Blockchains ». *101 Blockchains* [en ligne]. 28 mai 2020. [Consulté le 3 septembre 2022]. Disponible à l'adresse : Permissioned vs Permissionless Blockchains - 101 Blockchains

Annexe 1 : Code smart contract Javascript

Copier ce code pour remplacer entièrement la page assetTransfer.js

```
'use strict';

const { Contract } = require('fabric-contract-api');

class AssetTransfer extends Contract {

    async InitLedger(ctx) {
        const assets = [
            {ID: 'asset1',Title: 'Harry Potter et la coupe de feu',Price: 25,Owner: 'Orga1',Quantity:
30,},
            {ID: 'asset2',Title: 'Game of Thrones 1',Price: 35,Owner: 'Orga1',Quantity: 40,},
            {ID: 'asset3',Title: 'Le Seigneur des Anneaux',Price: 30,Owner: 'Orga1',Quantity: 15,},
            {ID: 'asset4',Title: 'Le clan des Otoris',Price: 15,Owner: 'Orga2',Quantity: 3,},
            {ID: 'asset5',Title: 'Fondation',Price: 15,Owner: 'Orga2',Quantity: 5,},
            {ID: 'asset6',Title: 'Dune',Price: 20,Owner: 'Orga2',Quantity: 15,},
        ];

        for (const asset of assets) {
            asset.docType = 'asset';
            await ctx.stub.putState(asset.ID, Buffer.from(JSON.stringify(asset)));
            console.info(`Asset ${asset.ID} initialized`);
        }
    }

    async CreateAsset(ctx, id, title, price, owner, quantity) {
        const asset = {
            ID: id,
            Title: title,
            Price: price,
            Owner: owner,
            Quantity: quantity,
        };
        ctx.stub.putState(id, Buffer.from(JSON.stringify(asset)));
        return JSON.stringify(asset);
    }

    async ReadAsset(ctx, id) {
        const assetJSON = await ctx.stub.getState(id);
        if (!assetJSON || assetJSON.length === 0) {
            throw new Error(`The asset ${id} does not exist`);
        }
        return assetJSON.toString();
    }

    async UpdateAsset(ctx, id, title, price, owner, quantity) {
        const exists = await this.AssetExists(ctx, id);
        if (!exists) {
            throw new Error(`The asset ${id} does not exist`);
        }

        const updatedAsset = {
            ID: id,
            Title: title,
            Price: price,
            Owner: owner,
            Quantity: quantity,
        };
        return ctx.stub.putState(id, Buffer.from(JSON.stringify(updatedAsset)));
    }

    async DeleteAsset(ctx, id) {
        const exists = await this.AssetExists(ctx, id);
        if (!exists) {
            throw new Error(`The asset ${id} does not exist`);
        }
        return ctx.stub.deleteState(id);
    }

    async AssetExists(ctx, id) {
        const assetJSON = await ctx.stub.getState(id);
        return assetJSON && assetJSON.length > 0;
    }
```

```
    async TransferAsset(ctx, id, newOwner) {
        const assetString = await this.ReadAsset(ctx, id);
        const asset = JSON.parse(assetString);
        asset.Owner = newOwner;
        return ctx.stub.putState(id, Buffer.from(JSON.stringify(asset)));
    }

    async GetAllAssets(ctx) {
        const allResults = [];
        const iterator = await ctx.stub.getStateByRange('', '');
        let result = await iterator.next();
        while (!result.done) {
            const strValue = Buffer.from(result.value.value.toString()).toString('utf8');
            let record;
            try {
                record = JSON.parse(strValue);
            } catch (err) {
                console.log(err);
                record = strValue;
            }
            allResults.push({ Key: result.value.key, Record: record });
            result = await iterator.next();
        }
        return JSON.stringify(allResults);
    }
}
module.exports = AssetTransfer;
```

Annexe 2 : Code application Javascript

```
'use strict';

const { Gateway, Wallets } = require('fabric-network');
const FabricCAServices = require('fabric-ca-client');
const path = require('path');
const { buildCAClient, registerAndEnrollUser, enrollAdmin } = require('../../test-
application/javascript/CAUtil.js');
const { buildCCPOrg1, buildWallet } = require('../../test-application/javascript/AppUtil.js');

const channelName = 'mychannel';
const chaincodeName = 'basic';
const mspOrg1 = 'Org1MSP';
const walletPath = path.join(__dirname, 'wallet');
const org1UserId = 'appUser';

function prettyJSONString(inputString) {
	return JSON.stringify(JSON.parse(inputString), null, 2);
}

async function main() {
	try {
		const ccp = buildCCPOrg1();
		const caClient = buildCAClient(FabricCAServices, ccp, 'ca.org1.example.com');
		const wallet = await buildWallet(Wallets, walletPath);
		await enrollAdmin(caClient, wallet, mspOrg1);
		await registerAndEnrollUser(caClient, wallet, mspOrg1, org1UserId,
'org1.department1');

		const gateway = new Gateway();

		try {
			await gateway.connect(ccp, {
				wallet,
				identity: org1UserId,
				discovery: { enabled: true, asLocalhost: true }
			});

			const network = await gateway.getNetwork(channelName);

			const contract = network.getContract(chaincodeName);

			console.log('\n--> Submit Transaction: InitLedger, function creates
the initial set of assets on the ledger');
			await contract.submitTransaction('InitLedger');
			console.log('*** Result: committed');

			console.log('\n--> Evaluate Transaction: GetAllAssets, function
returns all the current assets on the ledger');
			let result = await contract.evaluateTransaction('GetAllAssets');
			console.log(`*** Result: ${prettyJSONString(result.toString())}`);

			console.log('\n--> Submit Transaction: CreateAsset, creates new asset
');
			result = await contract.submitTransaction('CreateAsset', 'asset13',
'L\'homme Bicentenaire', '15', 'Orga2', '10');
			console.log('*** Result: committed');
			if (`${result}` !== '') {
				console.log(`*** Result:
${prettyJSONString(result.toString())}`);
			}

			console.log('\n--> Evaluate Transaction: ReadAsset, function returns
an asset with a given assetID');
			result = await contract.evaluateTransaction('ReadAsset', 'asset13');
			console.log(`*** Result: ${prettyJSONString(result.toString())}`);

			console.log('\n--> Evaluate Transaction: AssetExists, function
returns "true" if an asset with given assetID exist');
			result = await contract.evaluateTransaction('AssetExists', 'asset1');
			console.log(`*** Result: ${prettyJSONString(result.toString())}`);

			console.log('\n--> Submit Transaction: UpdateAsset asset1, change the
quantity from 30 to 20');
			await contract.submitTransaction('UpdateAsset', 'asset1', 'Harry
Potter et la coupe de feu', 25, 'Orga1', 20);
```

```
                                console.log('*** Result: committed');

                                console.log('\n--> Evaluate Transaction: ReadAsset, function returns
"asset1" attributes');
                                result = await contract.evaluateTransaction('ReadAsset', 'asset1');
                                console.log(`*** Result: ${prettyJSONString(result.toString())}`);

                                console.log('\n--> Submit Transaction: TransferAsset asset1, transfer
to new owner from Orga1 to Orga2');
                                await contract.submitTransaction('TransferAsset', 'asset1', 'Orga2');
                                console.log('*** Result: committed');

                                console.log('\n--> Evaluate Transaction: ReadAsset, function returns
"asset1" attributes');
                                result = await contract.evaluateTransaction('ReadAsset', 'asset1');
                                console.log(`*** Result: ${prettyJSONString(result.toString())}`);
                        } finally {
                                gateway.disconnect();
                                console.log('Application terminated');
                        }
                } catch (error) {
                        console.error(`******** FAILED to run the application: ${error}`);
                }
}

main();
```

Annexe 3 : Résultat étape 1 et 2 de l'application Javascript

```
--> Submit Transaction: InitLedger, function creates the initial set of assets on the ledger
*** Result: committed

--> Evaluate Transaction: GetAllAssets, function returns all the current assets on the ledger
*** Result: [
  {
    "Key": "asset1",
    "Record": {
      "ID": "asset1",
      "Title": "Harry Potter et la coupe de feu",
      "Price": 25,
      "Owner": "Orga1",
      "Quantity": 30,
      "docType": "asset"
    }
  },
  {
    "Key": "asset2",
    "Record": {
      "ID": "asset2",
      "Title": "Game of Thrones 1",
      "Price": 35,
      "Owner": "Orga1",
      "Quantity": 40,
      "docType": "asset"
    }
  },
  {
    "Key": "asset3",
    "Record": {
      "ID": "asset3",
      "Title": "Le Seigneur des Anneaux",
      "Price": 30,
      "Owner": "Orga1",
      "Quantity": 15,
      "docType": "asset"
    }
  },
  {
    "Key": "asset4",
    "Record": {
      "ID": "asset4",
      "Title": "Le clan des Otoris",
      "Price": 15,
      "Owner": "Orga2",
      "Quantity": 3,
      "docType": "asset"
    }
  },
  {
    "Key": "asset5",
    "Record": {
      "ID": "asset5",
      "Title": "Fondation",
      "Price": 15,
      "Owner": "Orga2",
      "Quantity": 5,
      "docType": "asset"
    }
  },
  {
    "Key": "asset6",
    "Record": {
      "ID": "asset6",
      "Title": "Dune",
      "Price": 20,
      "Owner": "Orga2",
      "Quantity": 15,
      "docType": "asset"
    }
  },
```

1

2