



# La technologie de Blockchain



Akram Hakiri (akram,hakiri@gmail.com)

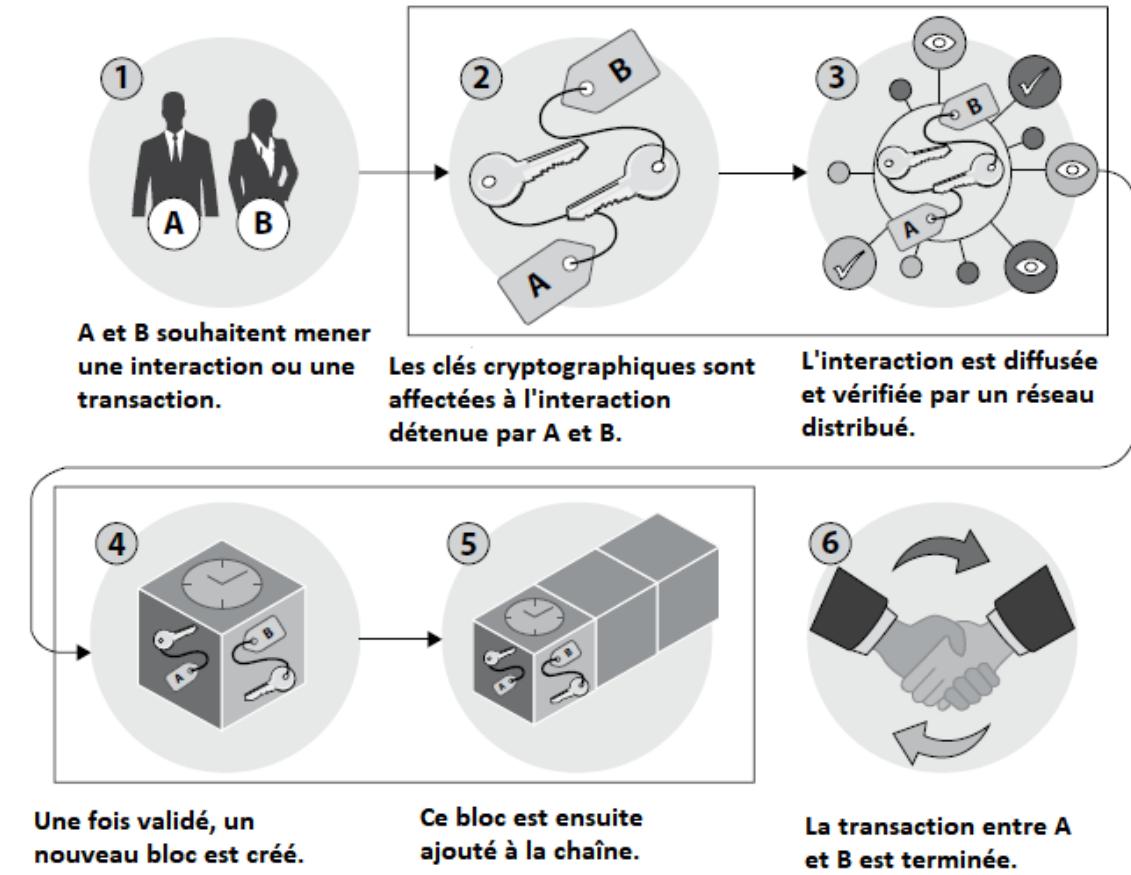
# Aperçu de la présentation

---

- ▶ **Introduction à la Blockchain**
- ▶ **Cryptographie dans la blockchain**
- ▶ **Plateformes Blockchain**
  - ▶ Bitcoin, Ethereum, Hyperledger,
- ▶ **Impact industriel et cas d'utilisation**
  - ▶ Finance, immobilier, assurance, gouvernement, technologies IoT
- ▶ **Introduction à la blockchain Ethereum & Solidity**
- ▶ **Atelier pratiques**
  - ▶ Atelier 1: Application Tirage au sort dans la blockchain
  - ▶ Atelier 2 : Développer votre propre crypto-monnaie virtuelle
  - ▶ Atelier 3 : Développer une application blockchain décentralisée (DApps)
  - ▶ Atelier 4 : Développer une application IoT sécurisée dans la blockchain (Concepts avancés)
  - ▶

# Qu'est ce que c'est la Blockchain

- ▶ Une blockchain est une base de données englobant une chaîne physique de blocs de longueur fixe comprenant 1 à N transactions,
- ▶ Chaque transaction ajoutée à un nouveau bloc étant validée puis insérée dans le bloc.
- ▶ Lorsque le bloc est terminé, il est ajouté à la fin de la chaîne de blocs existante.
- ▶ Seulement deux opérations (contrairement au CRUD classique) consistent à ajouter une transaction et à afficher une transaction,

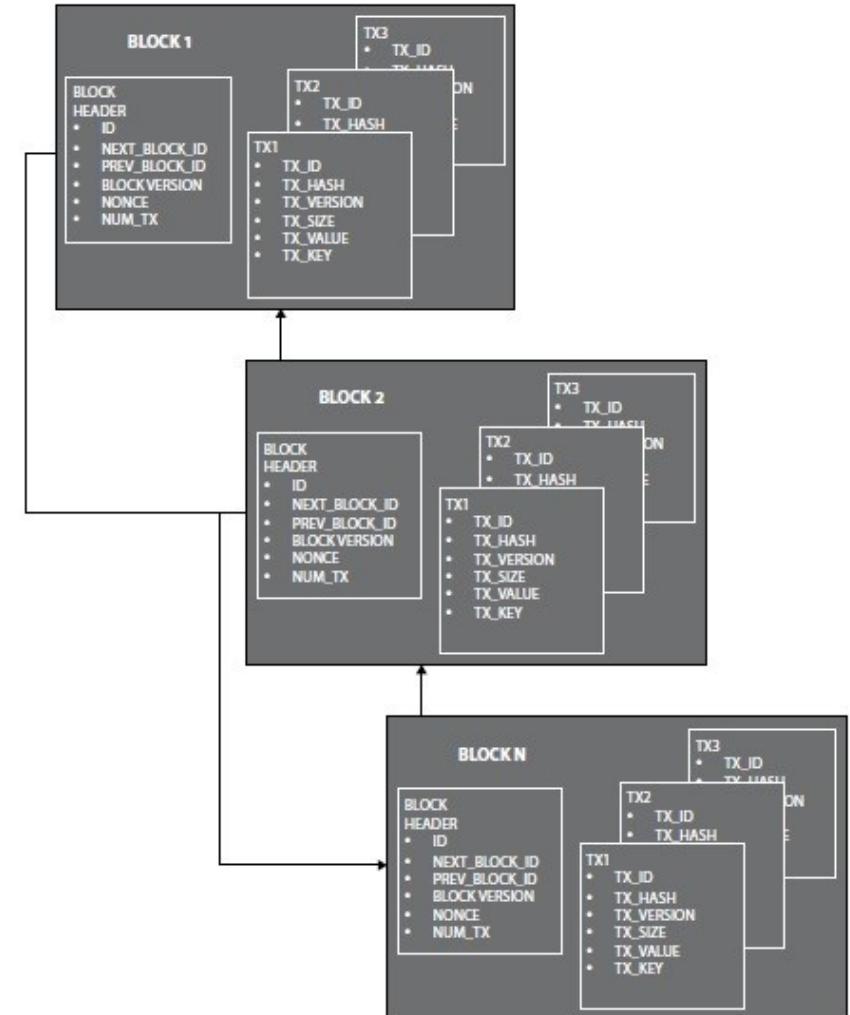


# Qu'est ce que c'est la Blockchain

- ▶ Une Blockchain est une **base de données distribuée** qui gère une **liste doublement chainée liées de blocs ordonnées**

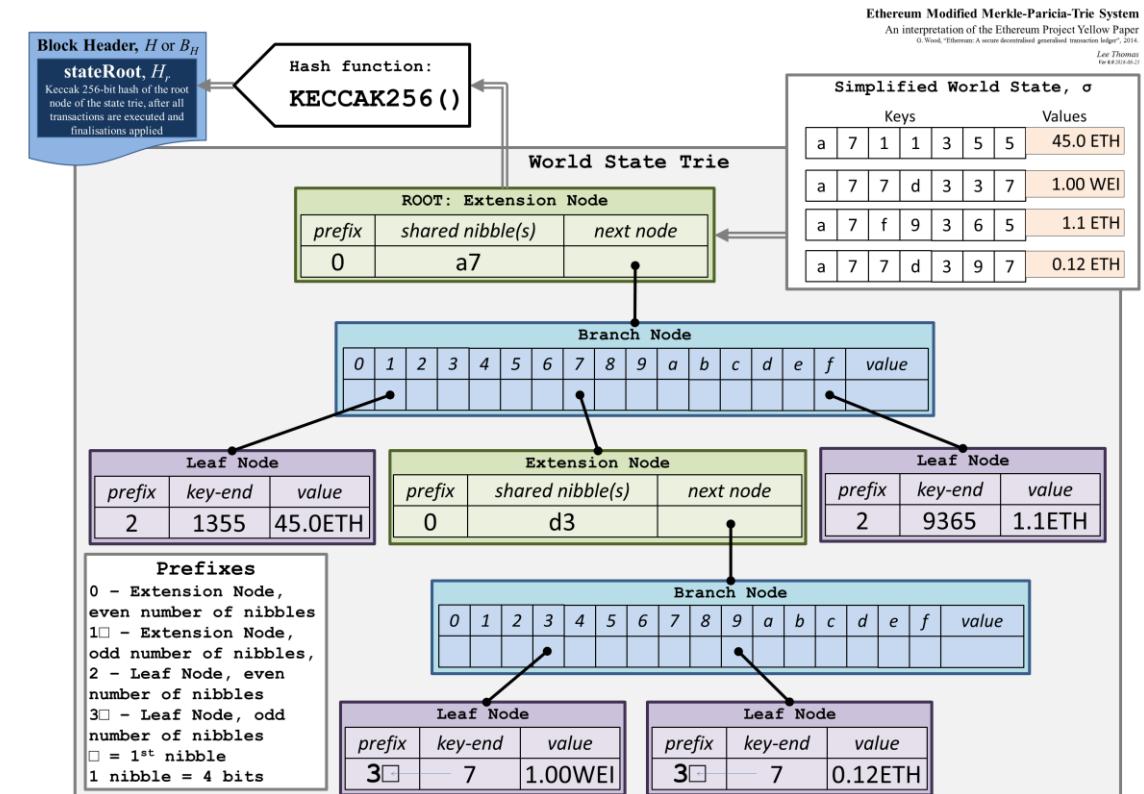
- ▶ Chaque bloc représente en moyenne 1 Mo mégaoctet et contient des données de contrôle d'environ 200 octets, telles qu'un horodatage, un lien vers un bloc précédent, d'autres champs

<https://www.blockchain.com/charts/avg-block-size>



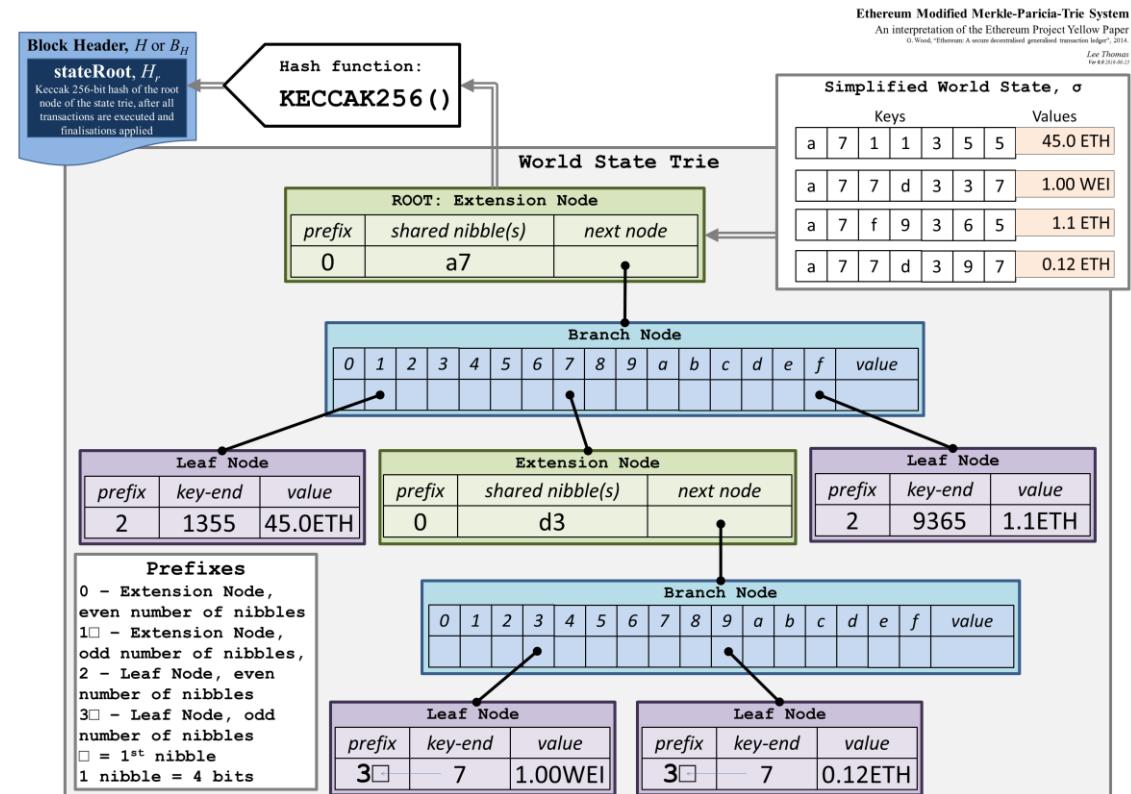
# Notion de Block

- ▶ Une blockchain est un registre distribué de transactions
- ▶ Les transactions sont implémentées sous forme de données groupées en blocs,
- ▶ Les transactions utilisent une validation cryptographique pour lier les blocs entre eux.
- ▶ Chaque bloc référence et identifie le bloc précédent en utilisant une fonction de hachage qui forme une chaîne ininterrompue (c'est-à-dire, une chaîne de blocs).
- ▶ <https://anders.com/blockchain/blockchain.html>



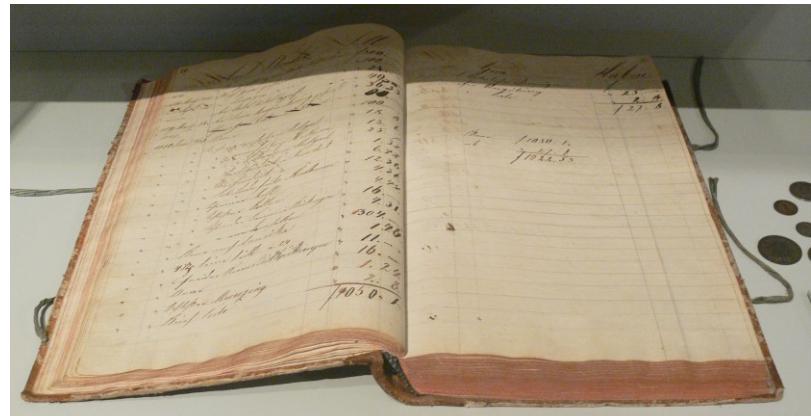
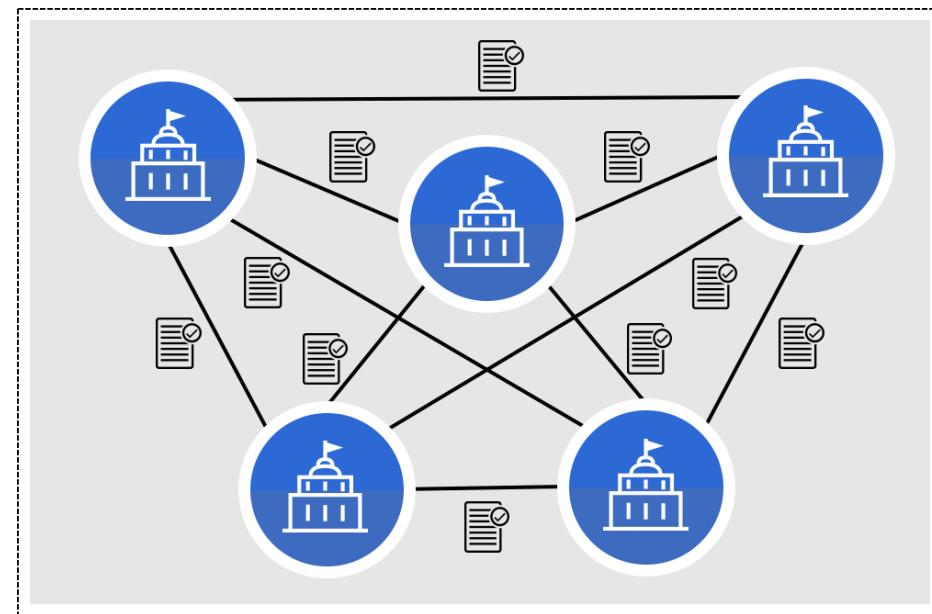
# Structure d'une blockchain

- ▶ Une fois enregistrés, les **blocs sont conçus pour résister à toute modification**
- ▶ Les données d'un bloc ne peuvent pas être modifiées **rétroactivement**.
- ▶ Les nouvelles données sont ajoutées
  - ▶ aux anciennes, c.-à-d. les données sont uniquement **écrites, jamais supprimées**.
  - ▶ par **lots** ou par blocs, puis **ajoutées** aux blocs existants, **formant une chaîne** de blocs.
- ▶ Non seulement tout le monde a la même base de données (blockchain), mais tout le monde a une copie dans la blockchain auquel il est le seul à pouvoir accéder.



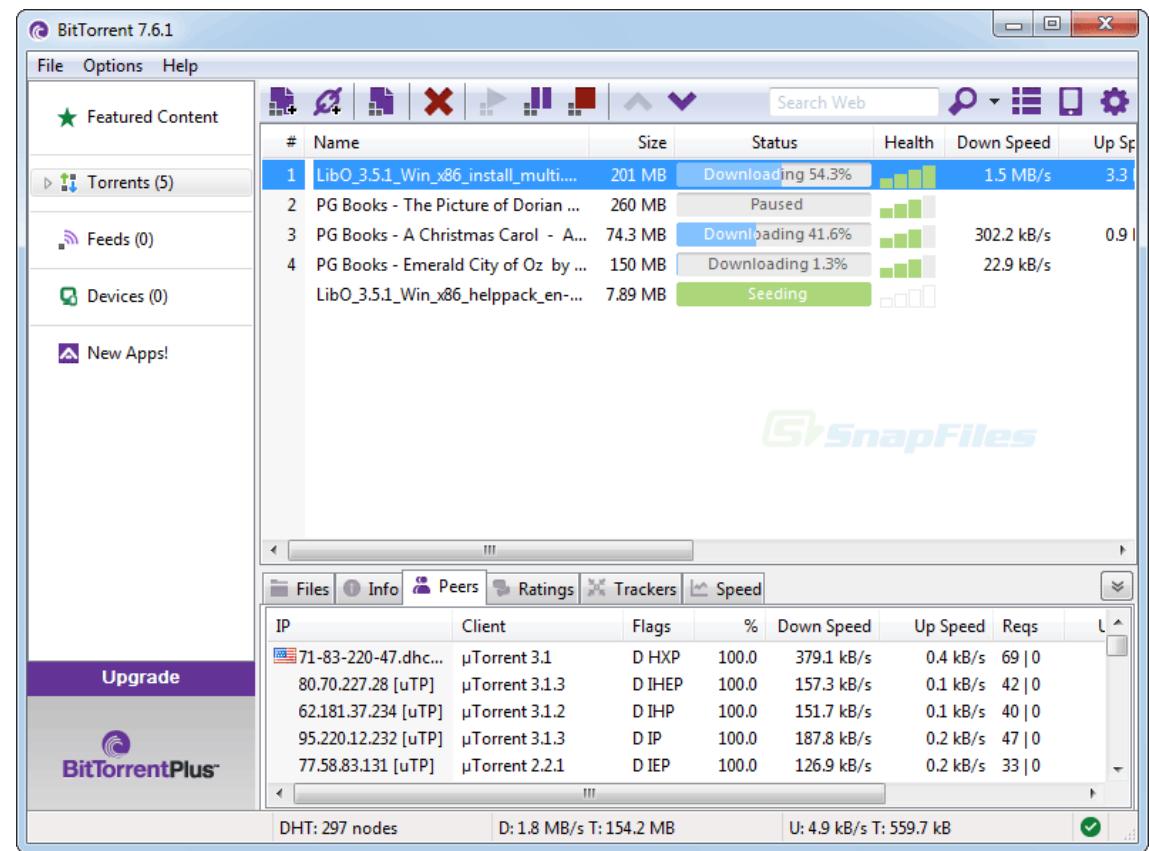
# Distributed Ledger

- ▶ Une blockchain est gérée de manière autonome grâce à un réseau distribué pair-à-pair (Peer2Peer) de parties indépendantes.
- ▶ Les Blockchains sont considérées comme un grand livre (**registre ou Ledger**) ouvert et distribué pouvant enregistrer les transactions de manière efficace, vérifiable et permanente.
- ▶ Le registre lui-même peut également être programmé pour déclencher automatiquement des transactions.



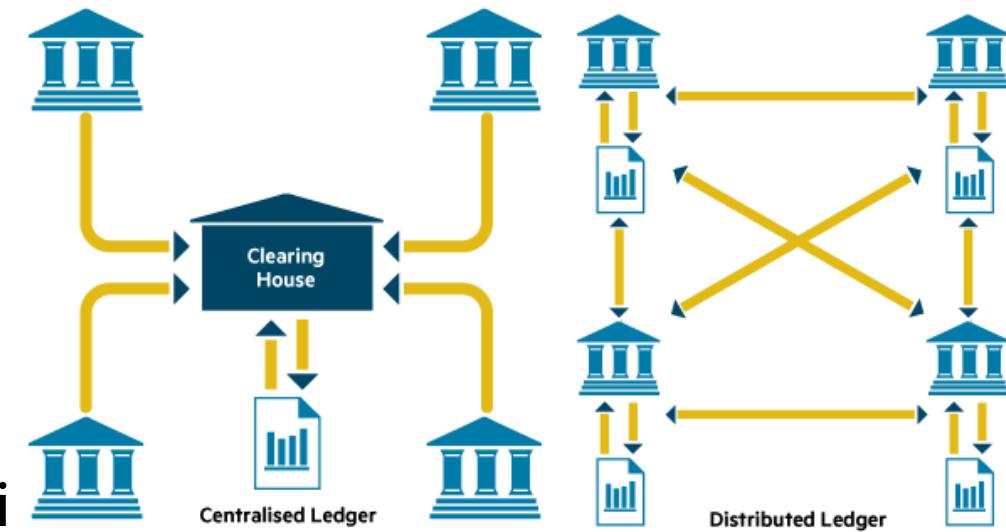
# Analogie avec BitTorrent

- ▶ Les technologies de blockchain permettent de réaliser une coopération à grande échelle de manière totalement distribuée et décentralisée.
- ▶ Ça nous rappelle les logiciels comme BitTorrent qui permet de télécharger des fichiers directement avec d'autres utilisateurs BitTorrent.



# Blockchain publique

- ▶ La blockchain publique n'a pas d'autorité centrale pour gérer les noms d'utilisateur et les mots de passe, elle utilise donc la cryptographie.
- ▶ Ainsi, bien que tout le monde puisse voir les données étiquetées avec votre adresse dans la blockchain, personne n'est autorisé à la modifier.
- ▶ Il ne peut être modifié que par la personne qui peut prouver qu'elle est le propriétaire via la clé privée.



# Adresses dans la blockchain

- ▶ Ce qui est également étonnant dans ce système, c'est que tout le monde peut générer une adresse par lui-même, de manière isolée, sans craindre qu'il ne se heurte à l'adresse d'un tiers.
- ▶ La raison est qu'il y a tellement d'adresses possibles qu'il est essentiellement impossible de se heurter à une autre adresse, même si vous avez essayé.
- ▶ <https://www.blockchain.com/fr/explorer>

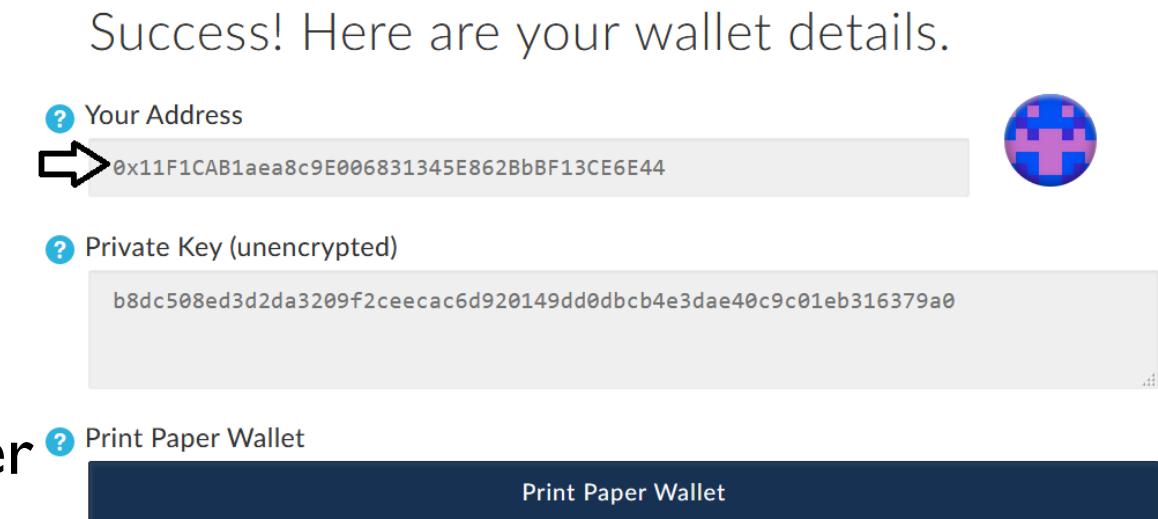
Success! Here are your wallet details.

Your Address  
0x11F1CAB1aea8c9E006831345E862BbBF13CE6E44

Private Key (unencrypted)  
b8dc508ed3d2da3209f2ceecac6d920149dd0dbcb4e3dae40c9c01eb316379a0

Print Paper Wallet

Print Paper Wallet



# Résumant la notion de la Blockchain

---

- ▶ La blockchain est une base de données distribuée, transparente, immuable, validée, sécurisée et pseudo-anonyme qui existe sous la forme de plusieurs nœuds, de sorte que si 51% des nœuds sont en accord, la confiance de la chaîne est garantie.
- ▶ La blockchain est distribuée car une copie complète réside sur autant de nœuds que dans le système.
- ▶ La blockchain est immuable car aucune des transactions ne peut être modifiée.
- ▶ La blockchain est validée (par exemple dans l'espace Bitcoin) par les mineurs indemnisés pour la construction du prochain bloc sécurisé.
- ▶ La blockchain est pseudo-anonyme car l'identité des personnes impliquées dans la transaction est représentée par une clé d'adresse sous la forme d'une chaîne aléatoire.

# Les transactions

---

- ▶ Les transactions sont des messages signés générés par un compte externe, transmis par le réseau et enregistrés sur la blockchain.
- ▶ La structure d'une transaction:
  - ▶ **Nonce** : numéro de séquence, émis par l'EOA d'origine, utilisé pour empêcher la relecture du message.
  - ▶ **Gaz price**: le prix du gaz (en wei) que l'initiateur est disposé à payer
  - ▶ **Gaz limit**: la quantité maximale de gaz que l'initiateur est disposé à acheter pour cette transaction
  - ▶ **Recipient** : L'adresse Ethereum de destination
  - ▶ **Value** : La quantité d'éther à envoyer à la destination
  - ▶ **Data** : La charge utile de données binaires de longueur variable
  - ▶ **v,r,s** : Les trois composants d'une signature numérique ECDSA de l'EOA d'origine

# La transaction Nonce

---

- ▶ La transaction Nonce est l'une des composantes les plus importantes
- ▶ Nonce :
  - ▶ valeur scalaire égale au nombre de transactions envoyées à partir de cette adresse
  - ▶ ou, dans le cas de comptes avec code associé, au nombre de créations de contrat effectuées par ce compte.
  - ▶ il est calculé dynamiquement, en comptant le nombre de transactions confirmées provenant d'une adresse.

# Types de Blockchain: Permissioned vs Permissionless

---

- ▶ **Blockchain publique (Permissioned)**
  - ▶ Une blockchain publique telle que bitcoin permet à quiconque de stocker, envoyer et recevoir des données. qui vous est fourni en tant que membre du réseau.
  - ▶ le consentement à lire et à écrire des données sur une blockchain publique est partagé équitablement entre tous les participants.
  - ▶ La blockchain publique n'offre aucun privilège spécial à qui que ce soit.
- ▶ **Blockchains privés (Permissionless)**
  - ▶ Une blockchain déployé par un acteur privé, par exemple, American Express a déployé sa propre blockchain Hyperledger
- ▶ **Blockchain de consortium**
  - ▶ La blockchain est partagée entre différents acteurs ayant un intérêt à collaborer ensemble.
  - ▶ Exemple:
    - ▶ la blockchain Corda <https://www.corda.net/> est partagée par différentes banques pour être utilisée pour du règlements,
    - ▶ Certaines banques ont quitté le projet pour des raisons concurrentielle
    - ▶ Blockchain fédérateur (partage des cout de maintenance), et devient centralisé

# Permissioned vs Permissionless Blockchains



# Implémentation de Blockchains

---

- ▶ Bitcoin <https://bitcoin.org>
- ▶ Ethereum <https://www.ethereum.org/>
- ▶ Namecoin: <https://namecoin.org/>
  - ▶ Crée en 2010, Namecoin est une technologie open-source décentralisée axée sur la résistance à la censure (Proof of Work). <https://github.com/namecoin>
  - ▶ Elle utilise un serveur DNS alternatif « dot,bit » qui permet l'enregistrement de noms de domaines décentralisés.
- ▶ Ripple: <https://ripple.com/>
  - ▶ Ripple est un start-up de San Francisco qui a conçu un système de paiement similaire au Blockchain. <https://github.com/ripple>
  - ▶ Utilise un protocole de paiement qui fonctionne de manière similaire à un système de paiement, un réseau de remise et un échange de devises.
  - ▶ Il fonctionne avec les crypto monnaie (ex, Banque Nationale d'Abu Dhabi)

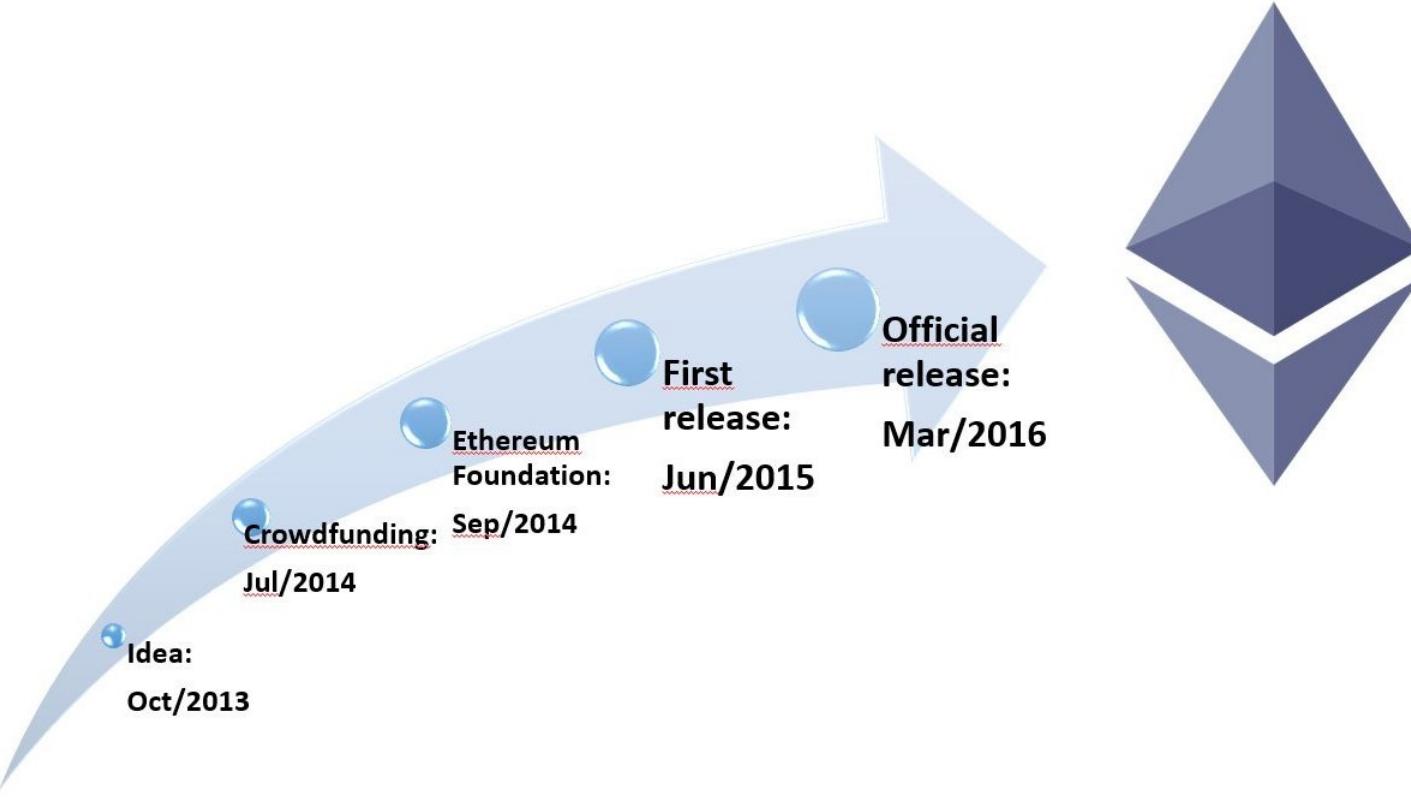
# Bitcoin

---

- ▶ Bitcoin est une monnaie décentralisée, mise en pratique pour la première fois par Satoshi Nakamoto en 2008.
- ▶ Bitcoin combine :
  - ▶ des primitives établies pour la gestion de la propriété (ownership) via la cryptographie à clé publique [https://www.blockchain.com/fr/api/api\\_receive](https://www.blockchain.com/fr/api/api_receive)
  - ▶ un algorithme de consensus permettant de garder la trace du propriétaire des pièces de monnaie, appelé preuve de travail (proof-of-work).
- ▶ Le mécanisme de preuve de travail joue deux rôles:
  - ▶ fournir un algorithme de consensus efficace, permettant aux nœuds du réseau de s'accorder collectivement sur un ensemble de mises à jour de l'état du registre Bitcoin.
  - ▶ Sécuriser la blockchain contre les cyberattaques, telles que les attaques par déni de service distribué (DDoS), en épuisant les ressources d'un système en envoyant plusieurs fausses demandes.

# Ethereum - Timeline

- ▶ Programmeur russe-canadien
- ▶ Ethereum co-fondé à l'âge de 19 ans



## Successful miner's computer

Takes the following steps and broadcasts block header,  $H$ , to network

### Determine Transactions

Miner picks transactions to process (from those broadcasted)

### Determine Ommers

Miner finds and includes valid ommers

### Apply Rewards

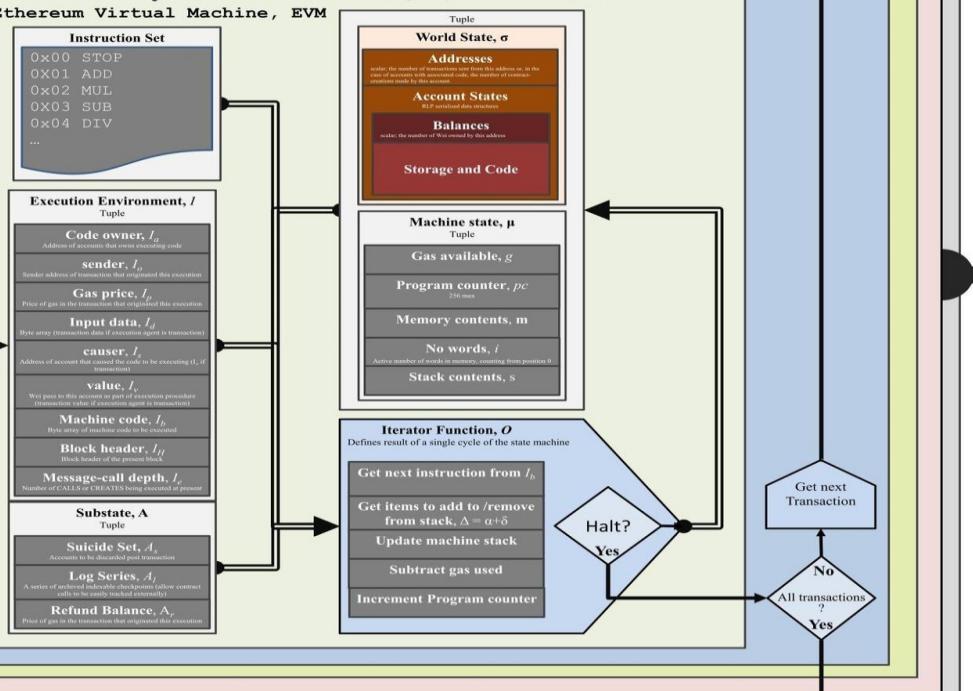
Update account balance(s) to reward valid blocks

### Compute a Valid State

**Block Finalisation** Defines result of all selected state transitions

**State Transition Cycle** Defines result of a single transaction  $\sigma_{t+1} = Y(\sigma_t, T)$

**Execution Cycle** Defines result of a single cycle of the state machine



### Proof of Work

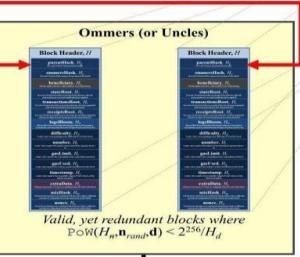
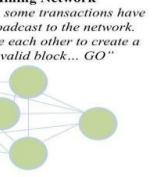
$\text{PoW}(H_n, n_{\text{rand}}, d) \leq 2^{256}/H_d$ ?

Choose Random Nonce  $n_{\text{rand}}$

8 bytes

## Mining Network

"Oh look, some transactions have been broadcast to the network. Let's race each other to create a new valid block... GO!"



## Ethereum Blockchain Mechanism (Proof Of Work)

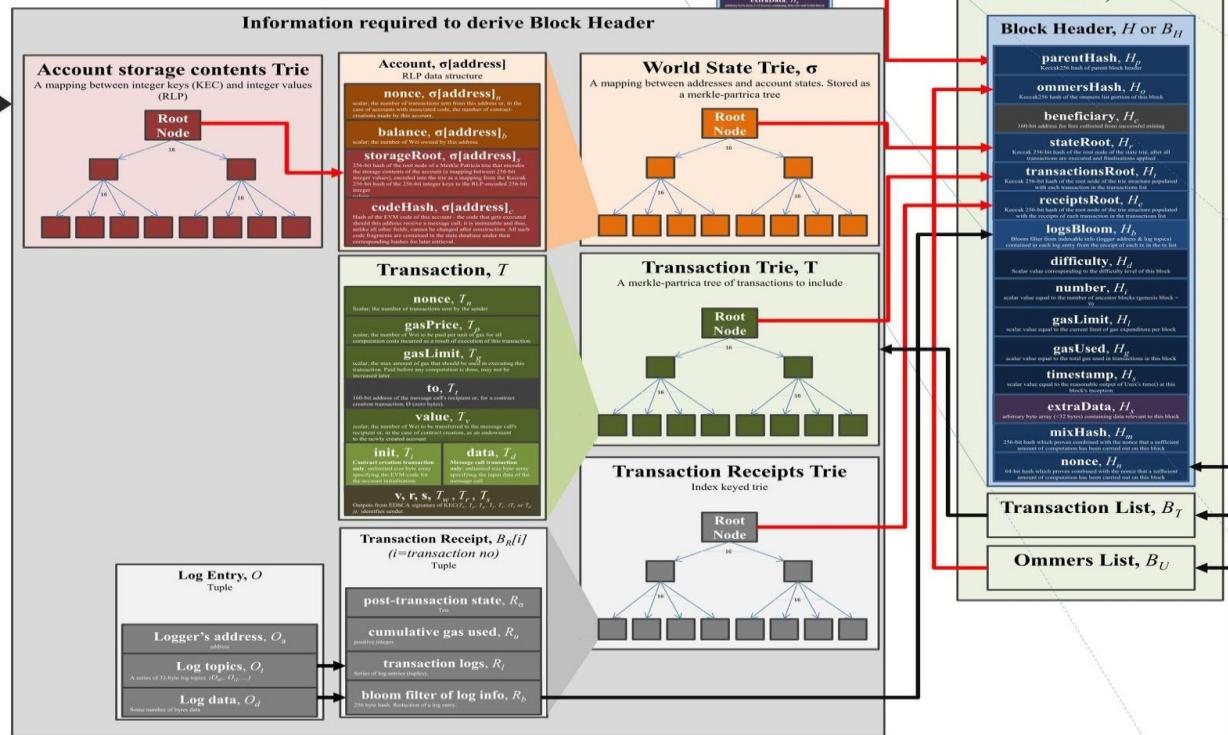
An interpretation of the Ethereum Project Yellow Paper  
G. Wood, "Ethereum: A secure decentralised generalised transaction ledger", 2014.

Lee Thomas  
2016-06-21  
Ver 0.1 2016-06-22

### Ethereum Network

"Oh look another miner has been successful. Therefore there is little point in continuing with this block – we'll start on the next one!"

"note – we had to run all the same code on our copies of the EVM to prove this!"



# Hyperledger

- ▶ Hyperledger est un effort collaboratif open source créé pour faire progresser les technologies de blockchain intersectorielles.

The screenshot shows the official website for Hyperledger, which is part of The Linux Foundation. The top navigation bar includes links for Members, Projects, Community, Resources, News & Events, Blog, About, and social media icons for YouTube, Twitter, Facebook, LinkedIn, and GitHub. Below the navigation is a search bar. The main content area features three news cards:

- Join The 2019 Hyperledger Internship Program**: A green card with text about the internship program and a "LEARN MORE" button.
- New Case Study: OrgBook Is Helping Cut Red Tape For Businesses**: A blue card with text about OrgBook and a "READ MORE" button.
- Hyperledger Welcomes 9 New Members to Its Enterprise Blockchain Community**: A purple card with text about new members joining the community and a "READ MORE" button.

## 5 projets et 3 outils

- Burrow - basé sur la machine virtuelle Ethereum & modulaire
- Fabric - Base pour le développement d'applications de chaîne de blocs modulaire
- Iroha - conçu pour les projets d'infrastructure
- Indy - Outils, bibliothèques et composants pour les identités
- Sawtooth - Preuve du temps écoulé et excellent pour IoT
- Cello - Créer, gérer et terminer
- Explorer - Afficher et surveiller (similaire à blockchain.info)
- Compositeur - Construisez des réseaux d'entreprise, aidez à rédiger des contrats intelligents, déployez

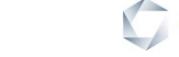
# Membres de la fondation Hyperledger



# Outils Open Source pour Hyperledger

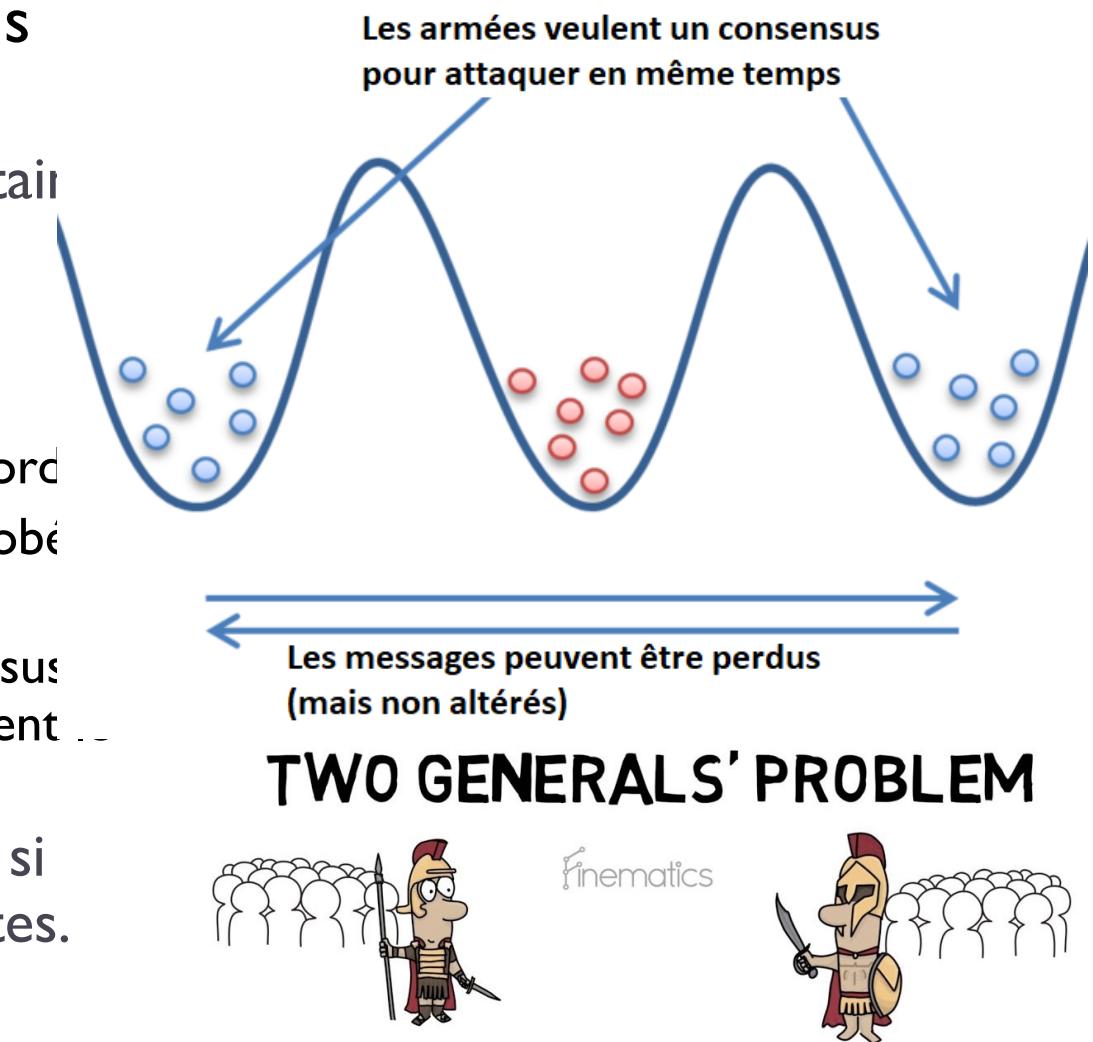


# Utilisation de blockchain Hyperledger en entreprises

Trade Finance	Pre and Post Trade	Complex Risk Coverage
Digital Trade Chain   	<b>DTCC</b>  	  
Identity/ Know your customer (KYC)	Unlisted Securities/ Private Equity Funds	Loyalty Program
<b>SECURE KEY</b>  	<b>NORTHERN TRUST</b>  	
Medicated Health Data Exchange	Fraud/ Compliance Registry	Distributed Energy/ Carbon Credit
		 
Supply Chain	Food Safety	Provenance/ Traceability
 	        	

# Problème des généraux byzantins

- ▶ Problème des généraux byzantins est plus générique (1982):
  - ▶ On peut avoir plusieurs généraux, dont certains peuvent être des traitres
  - ▶ un général doit donner un ordre à ses  $n-1$  lieutenants généraux :
    - ▶ tous les lieutenants fidèles obéissent au même ordre
    - ▶ Si le général est fidèle, chaque lieutenant fidèle obéit à l'ordre
    - ▶ Si le général commandant est traitre, un consensus doit être atteint, càd, tous les lieutenants prennent un vote à la majorité.
- ▶ Un algorithme peut atteindre un consensus si les deux tiers ( $2/3$ ) des acteurs sont honnêtes.

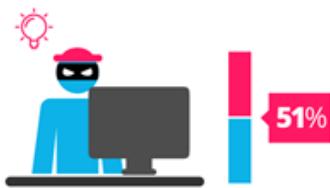


# Comment tout cela est-il lié à la blockchain?

## **Proof of Work**



"Proof of Work" est nécessaire pour définir un calcul informatique coûteux, appelé mining



une récompense est donnée au premier mineur qui résout chaque problème de bloc



les mineurs se font concurrence pour être le premier à trouver une solution au problème mathématique

## **vs Proof of Stake**



Proof of Stake, le créateur d'un nouveau bloc est choisi de manière déterministe, en fonction de sa richesse, i.e. Stake



Pour le PoS, pas de récompense globale, les mineurs prennent les frais de transaction



en PoS, Les crypto-monnaies peuvent être beaucoup plus rentables

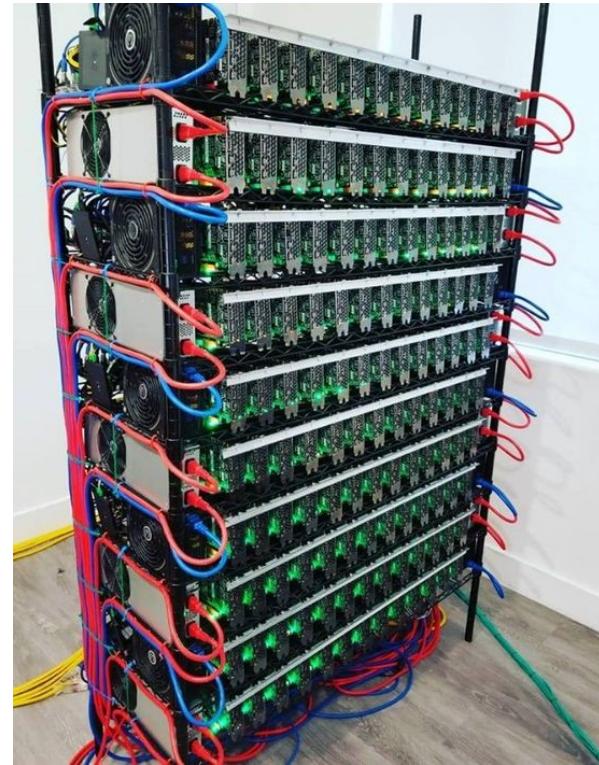
- ▶ Les blockchains sont des registres complètement décentralisés, sans aucune autorité centrale de contrôle
  - ▶ Comment le réseau peut-il être sûr que la transaction est valide?
  - ▶ Si quelqu'un n'essaie pas de dépenser deux fois le même argent.
- ▶ Un algorithme de consensus de type BFT est indispensable.
  - ▶ Bitcoin implémente une variante probabiliste de consensus, Proof-of-Work
  - ▶ Ethereum implémente une autre variante de consensus, Proof-of-Stake

# Concept du Proof-of-Work

---

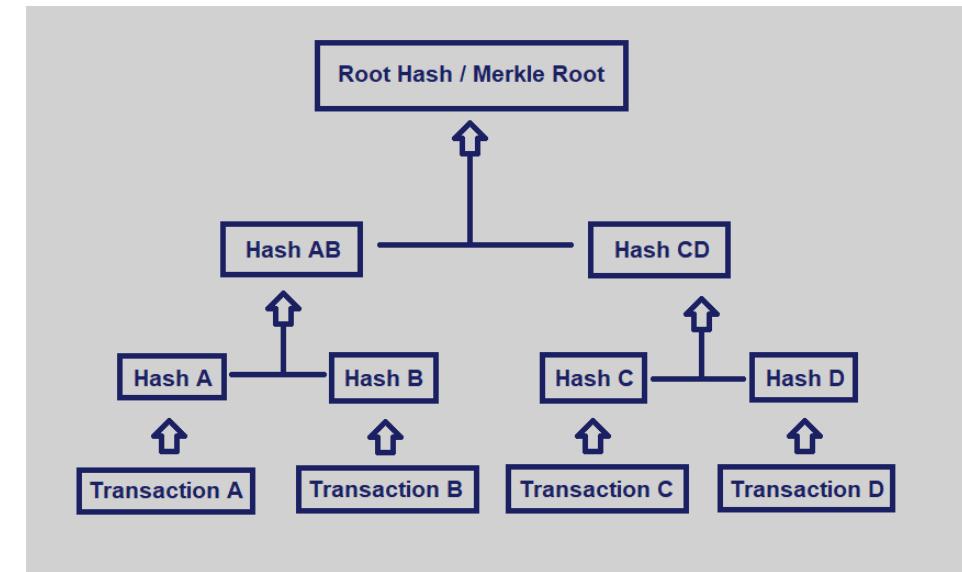
- ▶ Il définit un calcul Informatique intensif, dit « mining », pour créer des blocks sans confiance dans le registre de la blockchain
- ▶ Le terme « mining » a deux objectifs:
  - ▶ Vérifier la légitimité d'une transaction, ou pour éviter ce que l'on appelle une double dépense (attaque à 51%, aussi appelé double spending);
  - ▶ Créer de nouvelles monnaies numériques en récompensant les mineurs capable de créer des blocs plus rapidement que tous les autres mineurs (trouver des hash plus rapidement)
- ▶ Double Spending:
  - ▶ une attaque où il est nécessaire de posséder au moins 50% de la puissance de calcul de minage du Bitcoin. C'est-à-dire qu'une personne doit être capable de miner plus que tous les autres mineurs,
  - ▶ La plus grande organisation de mineurs ne possède que 26% de la capacité totale de calcul du réseau <https://blockchain.info/fr/pools> <https://btc.com/>

# Exemple d'infrastructure de mineurs de blockchain



# Arbre de Merkle

- ▶ un arbre de Merkle ou arbre de hachage binaire est une structure de données contenant non pas une liste de toute les transactions, mais plutôt une empreinte numérique (Hash) de toutes les transactions sous forme d'arborescence
- ▶ La racine de l'arbre est le nœud le plus haut. Les nœuds du bas sont appelés des nœuds feuilles.
- ▶ Chaque nœud est simplement un hachage cryptographique d'une transaction.
- ▶ <https://anders.com/blockchain/blockchain.html>



# Cryptographie dans la blockchain

---

- ▶ La cryptographie, est une branche des mathématiques largement utilisée en sécurité informatique.
- ▶ La cryptographie peut également être utilisée
  - ▶ pour **prouver la connaissance d'un secret sans révéler ce secret** (une signature numérique),
  - ▶ pour **prouver l'authenticité de données** (avec des empreintes digitales, également appelées «hachages»).
- ▶ Ces types d'épreuves cryptographiques sont des outils mathématiques essentiels au fonctionnement de tous les systèmes blockchain

# Cryptographie à Asymétrique

---

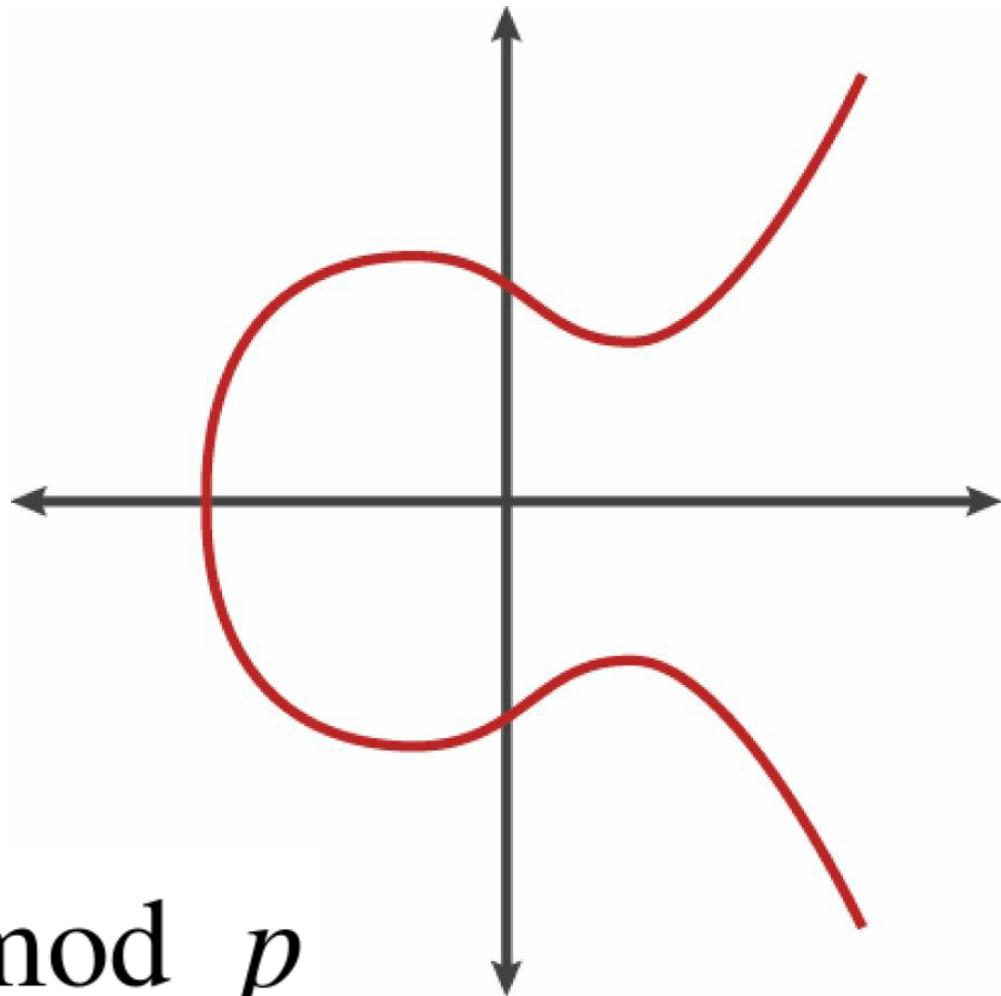
- ▶ C'est une est une méthode de chiffrement qui utilise deux clés (clé publique et une clé privée) qui se ressemblent mathématiquement mais qui ne sont pas identiques
- ▶ Elle utilise une catégorie avancée de fonctions mathématiques dites courbes elliptiques
- ▶ Il est impossible de deviner la clé privée à partir de la clé publique
- ▶ C'est pourquoi les clés publiques peuvent être partagées sans danger
- ▶ Le système de blockchain utilise la cryptographie asymétrique pour créer la clé publique et la clé privée

# Cryptographie à courbe elliptique

- ▶ Ethereum utilise une courbe elliptique spécifique et un ensemble de constantes mathématiques, comme défini dans une norme NIST appelée **secp256k1**.
- ▶ La courbe **secp256k1** est définie par la fonction suivante, qui produit une courbe elliptique:

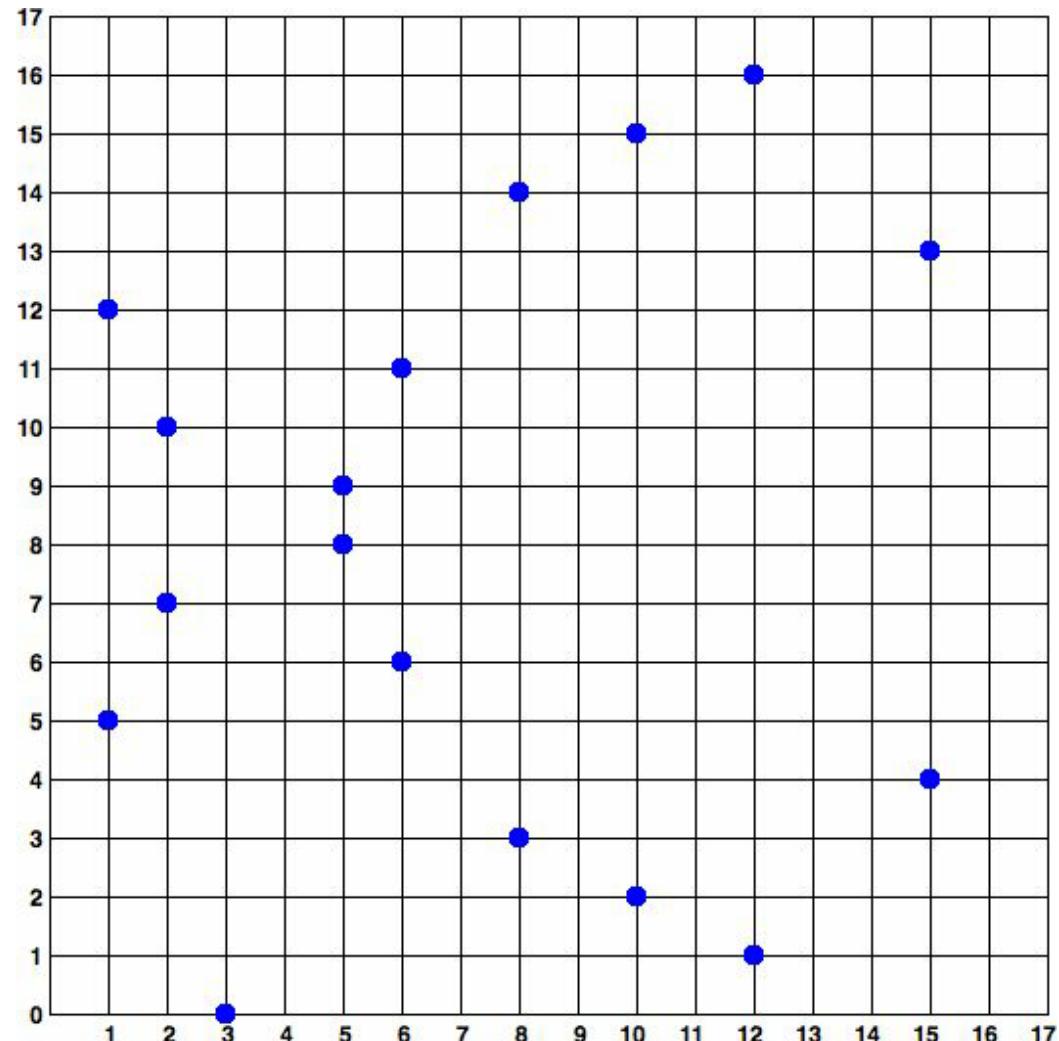
$$y^2 = (x^3 + 7) \text{ over } (\mathbb{F}_p)$$

$$y^2 \bmod p = (x^3 + 7) \bmod p$$



# Cryptographie à courbe elliptique

- ▶ visualisation d'une courbe elliptique sur  $F(p)$ , avec  $p = 17$
- ▶ Par exemple, un point  $Q$  avec des coordonnées  $(x, y)$  qui est un point sur la courbe **secp256k1**:
  - ▶  $Q = (x, y) = (4979039082524938448603314435591686, 4607616083520101638681403973749255, 924539515, 5957413216189990004586208649392101, 5780032175291755807399284007721050, 341297360)$
  - ▶ Les variables  $x$  et  $y$  sont les coordonnées du point  $Q$  sur la courbe

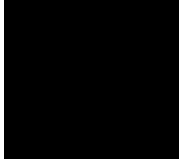


# Générer une clé publique

## ▶ Générer une clé publique

- ▶ Nous commençons par une clé privée sous la forme d'un nombre  $k$  générée aléatoirement,
- ▶ Nous la multiplions par un point prédéterminé de la courbe appelé point générateur  $G$  pour produire un autre point ailleurs sur la courbe, qui est la clé publique correspondante  $K$ :

$$K = k * G$$

 Clé Publique       Clé privée       Point Générateur

```
K = f8f8a2f43c8376ccb0871305060d7b27b0554d2cc72bccf41b2705608452f315 * G
```

# Fonctions de hachage

---

- ▶ Une fonction de hachage
  - ▶ est toute fonction pouvant être utilisée pour mapper des données de taille arbitraire à des données de taille fixe.
  - ▶ L'entrée dans une fonction de hachage est appelée une pré-image, le message ou simplement les données d'entrée.
  - ▶ La sortie s'appelle le hash.
- ▶ Les fonctions de hachage sont utilisées pour transformer des clés publiques en adresses de compte dans la blockchain, e.g. Ethereum.
- ▶ Ils peuvent également être utilisés pour créer des empreintes digitales facilitant la vérification des données.

# Fonctions de hachage cryptographique

---

- ▶ Une fonction de hachage cryptographique est une fonction de hachage unidirectionnelle qui mappe des données de taille arbitraire à une chaîne de bits de taille fixe.
- ▶ La nature «unidirectionnelle» signifie qu'il est impossible en termes de calcul de recréer les données d'entrée si on ne connaît que le hachage de sortie.
- ▶ Le seul moyen de déterminer une entrée possible consiste à effectuer une recherche par force brute, en vérifiant chaque candidat pour une sortie correspondante;
- ▶ Etant donné que l'espace de recherche est pratiquement infini, il est facile de comprendre l'impossibilité pratique de la tâche.

# Fonctions de hachage cryptographique

---

- ▶ Même si vous trouvez des données d'entrée qui créent un hachage correspondant, il se peut que ce ne soit pas les données d'entrée d'origine: les fonctions de hachage sont des fonctions «plusieurs à un».
- ▶ La recherche de deux ensembles de données en entrée qui aboutissent à la même sortie s'appelle rechercher une collision de hachage.
- ▶ En gros, plus la fonction de hachage est performante, plus les collisions de hachage sont rares. Pour Ethereum, ils sont effectivement impossibles.

## Fonction de hachage cryptographique d'Ethereum: Keccak-256

---

- ▶ En 2015, Ethereum utilise la fonction de hachage cryptographique Keccak-256, pour remplacer la norme SHA-3
- ▶ Par exemple, pour une entrée vide, on génère les sorties suivantes:
  - ▶ Keccak256("") =  
c5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470
  - ▶ SHA3("") =  
a7ffc6f8bf1ed76651c14756a061d662f580ff4de43b49fa82d80a4b80f8434a

# Les adresses Ethereum

---

- ▶ Les adresses Ethereum sont des identifiants uniques dérivés de clés publiques ou de contrats utilisant la fonction de hachage unidirectionnel Keccak-256.
- ▶ Exemple:
  - ▶ Clé privée k = f8f8a2f43c8376ccb0871305060d7b27b0554d2cc72bccf41b2705608452f315
  - ▶ Clé publique K =  
6e145cce1033dea239875dd00dfb4fee6e3348b84985c92f103444683bae07b83b5c38e5e ...
- ▶ Nous utilisons Keccak-256 pour calculer le hash de cette clé publique:
  - ▶ Keccak256(K) = 2a5bc342ed616b5ba5732269001d3f1ef827552ae1114027bd3ecf1f086ba0f9
- ▶ Ensuite, nous ne conservons que les 20 derniers octets (octets les moins significatifs), qui sont votre adresse Ethereum:
  - ▶ 001d3f1ef827552ae1114027bd3ecf1f086ba0f9
  - ▶ En hexadécimal: 0x001d3f1ef827552ae1114027bd3ecf1f086ba0f9

# Inter Exchange Client Address Protocol

---

- ▶ **Le protocole ICAP (Interchange Client Address Protocol)**
  - ▶ C'est un codage d'adresse Ethereum partiellement compatible avec le codage IBAN (International Bank Account Number),
  - ▶ Il offre un codage polyvalent, chiffré et interopérable pour les adresses Ethereum.
  - ▶ IBAN est un service centralisé et fortement réglementé.
  - ▶ ICAP est une implémentation décentralisée mais compatible pour les adresses Ethereum.
- ▶ Plus de détails sur le protocole ici:  
[https://github.com/ethereum/wiki/wiki/Inter-exchange-Client-Address-Protocol-\(ICAP\)](https://github.com/ethereum/wiki/wiki/Inter-exchange-Client-Address-Protocol-(ICAP))

## Hex Encoding with Checksum in Capitalization (EIP-55)

---

- ▶ En raison de la lenteur du déploiement d'ICAP et des services de noms, une norme a été proposée par la proposition d'amélioration Ethereum 55 (EIP-55).
- ▶ EIP-55 offre une somme de contrôle rétro-compatible pour les adresses Ethereum en modifiant la capitalisation de l'adresse hexadécimale.
- ▶ Plus de détails ici: <http://eips.ethereum.org/>

# Signatures numériques

---

- ▶ Les signatures numériques peuvent être utilisées pour présenter une preuve de propriété d'une clé privée sans révéler cette clé privée.
- ▶ Elle se base sur L'algorithme de signature numérique à courbe elliptique (ECDSA), qui est lui-même basé sur des paires clé privée-publique à courbe elliptique
- ▶ Une signature numérique remplit trois fonctions
  - ▶ Premièrement, la signature prouve que le propriétaire de la clé privée, qui est implicitement propriétaire d'un compte Ethereum, a autorisé la dépense d'éther ou l'exécution d'un contrat.
  - ▶ Deuxièmement, il garantit la non-répudiation: la preuve de l'autorisation est indéniable.
  - ▶ Troisièmement, la signature prouve que les données de la transaction n'ont pas été et ne peuvent être modifiées par personne après la signature de la transaction.

# Comment fonctionnent les signatures numériques

---

- ▶ Une signature numérique est un schéma mathématique composé de deux parties.
- ▶ La première partie est un algorithme permettant de créer une signature, à l'aide d'une clé privée (la clé de signature), à partir d'un message (qui dans notre cas est la transaction).
- ▶ La deuxième partie est un algorithme qui permet à quiconque de vérifier la signature en utilisant uniquement le message et une clé publique.

# Création d'une signature numérique

---

- ▶ Dans la mise en œuvre d'ECDSA par Ethereum, le «message» en cours de signature est la transaction, ou plus exactement le hachage Keccak-256 des données codées RLP de la transaction. La clé de signature est la clé privée de l'EOA. Le résultat est la signature:

$$Sig = F_{sig}(F_{keccak256}(m), k)$$

- ▶ où:
  - ▶ k est la clé privée de signature.
  - ▶ m est la transaction codée RLP.
  - ▶ Fkeccak256 est la fonction de hachage Keccak-256.
  - ▶ Fsig est l'algorithme de signature.
  - ▶ Sig est la signature résultante.
- ▶ La fonction Fsig produit une signature Sig composée de deux valeurs, communément appelées r et s:

$$Sig = (r, s)$$

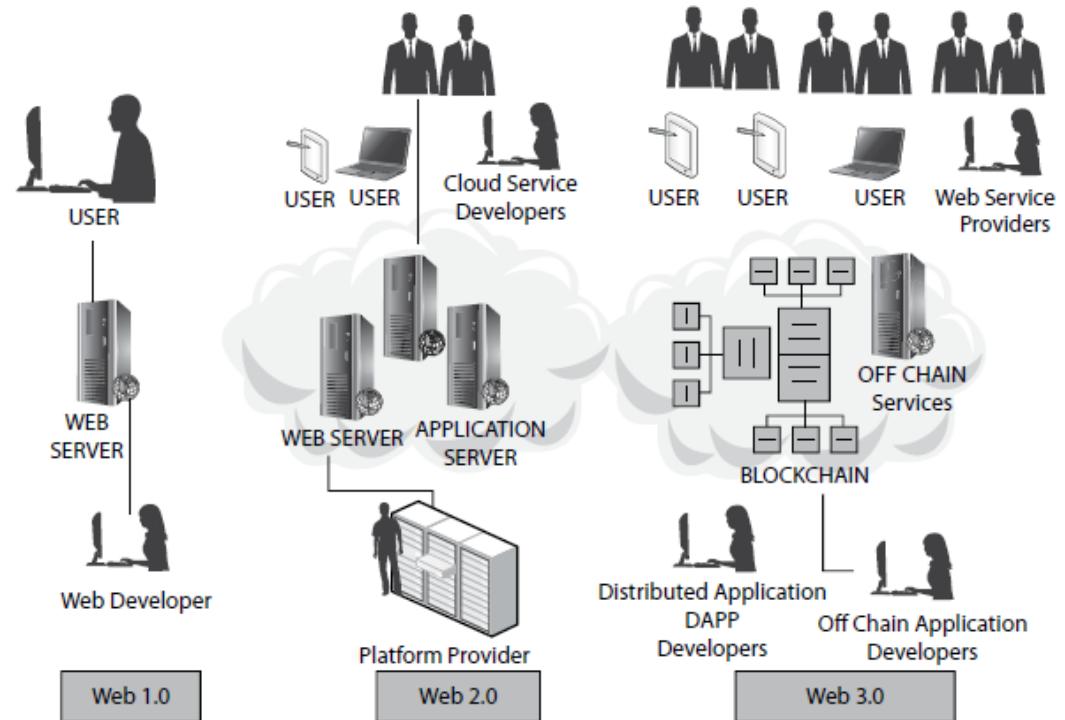


## Cas d'utilisation de Blockchain

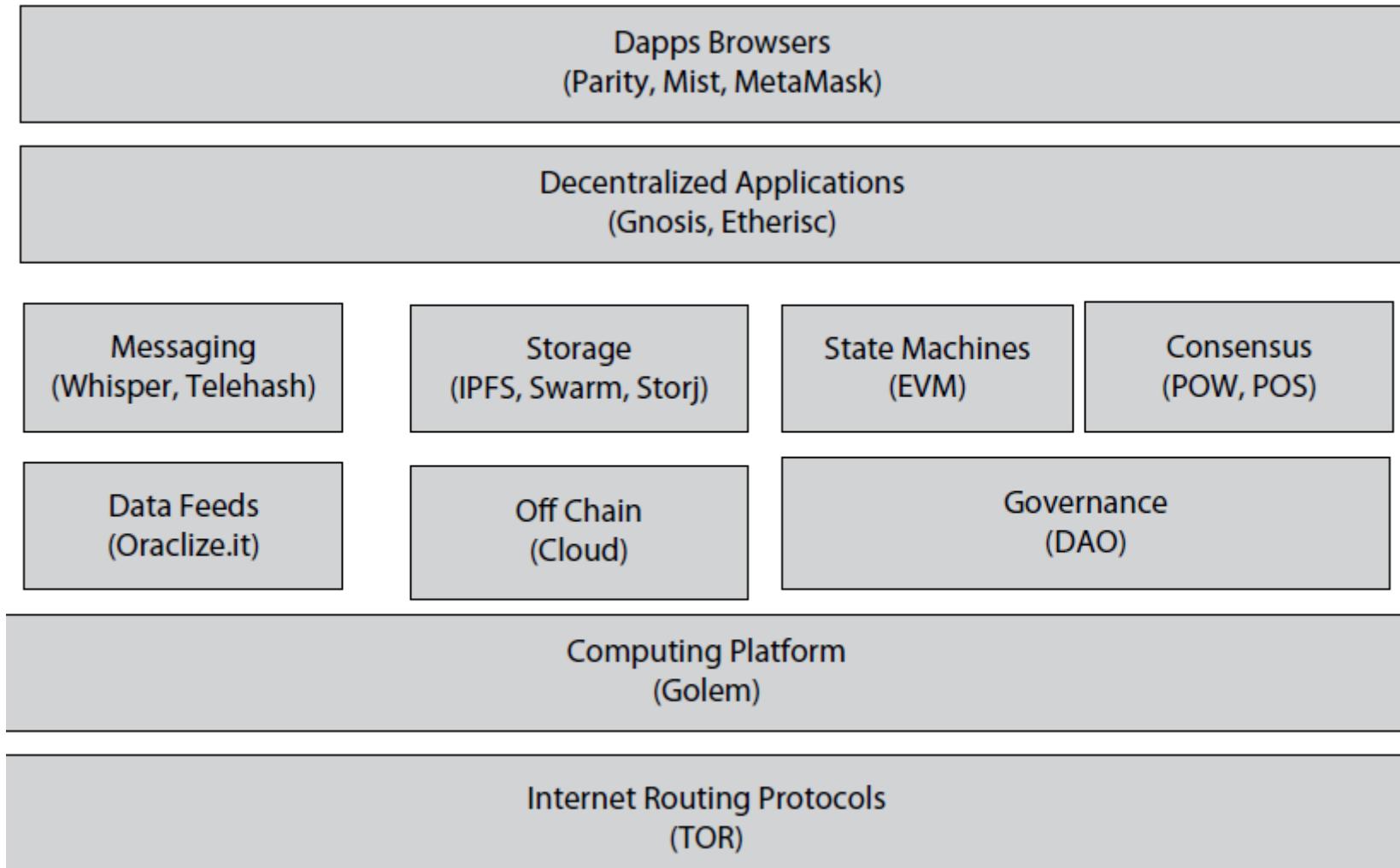


# Evolution vers le Web 3.0

- ▶ La technologie Blockchain marque le début de la nouvelle génération de service web
  - ▶ Complètement décentralisé
  - ▶ Aucun tier entre l'utilisateur et le service
  - ▶ Exemple: Bitcoin vs MasterCard



# Pile de protocole du Web 3.0



# Domaines d'application de la Blockchain

The screenshot shows the homepage of the Openledger website. At the top left is the logo "openledger". At the top right are navigation links: Products, Services, Solutions, Portfolio, About us, Insights, and a blue "Get in Touch" button. A dropdown menu titled "Solutions" is open, listing various application domains: Supply chain, Logistics, Real Estate, Automotive, Education, IoT, Healthcare, Insurance, and Retail. The background features a large white text "Enterprise block solutions developed" partially visible on the left, and a graphic on the right showing concentric circles and a stylized arrow pointing upwards.

Boost your business with blockchain-optimized processes, transparent operations and full immediate control over your assets

Get a Quote

Products Services Solutions Portfolio About us Insights Get in Touch

- Supply chain
- Logistics
- Real Estate
- Automotive
- Education
- IoT
- Healthcare
- Insurance
- Retail

<https://openledger.info/solutions/>

# Domaines d'application de la Blockchain

---

- ▶ **Cas d'utilisation professionnelle**
  - ▶ Crypto-monnaie et Jeton Numériques
  - ▶ Services bancaires et financiers
- ▶ **Cas d'utilisation technologique**
  - ▶ Systèmes de stockage distribués
  - ▶ Calcul distribué
  - ▶ Communications décentralisées
- ▶ **Cas d'utilisation juridique et de gouvernance**
  - ▶ Le début du droit autonome: un contrat intelligent
  - ▶ Organisations autonomes décentralisées

# La cryptomonnaie

---

- ▶ Plus de 2000 cryptomonnaies ont été introduite
  - ▶ Des nouvelles crypto monnaies apparaissent chaque jour et d'autres disparaissent
    1. Bitcoin (BTC)
    2. Ethereum (ETH)
    3. Ripple (XRP)
    4. Bitcoin Cash (BCH)
    5. Stellar Lumens (XLM)
    6. EOS (EOS)
    7. Litecoin (LTC)
    8. Cardano (ADA)
- ▶ <https://coinmarketcap.com/>
- ▶ Bitcoin était construit avec un stock limité à 21 millions de pièces.
  - ▶ Les pièces sont introduite progressivement sur le marché par un processus complexe, qui s'appelle « minage »
  - ▶ Le minage est le processus essentiel de validation, vérification, et engagement,
  - ▶ En 2018, 80% de stock de bitcoin a été introduite (sur 10 ans)

# Digital Tokens (Jeton numérique)

---

- ▶ Un jeton numérique peut être tout type d'actif numérique ou toute représentation numérique d'un actif physique.
- ▶ Dans Ethereum, un jeton numérique peut représenter n'importe quel produit fongible et échangeable, tel que
  - ▶ de la monnaie,
  - ▶ des points de fidélité,
  - ▶ des cheque-cadeaux, etc.

# Digital Tokens

---

- ▶ Tous les jetons basés sur Ethereum implémentent la fonctionnalité de base de manière standard, appelée norme de jeton ERC20
- ▶ L'importance des jetons numériques a considérablement augmenté avec les investissements de capitaux à risque et les investisseurs institutionnels
  - ▶ Une nouvelle méthode, appelée **offre initiale de pièces de monnaie** (issue de l'introduction en bourse, **appel public à l'épargne**), offrait un autre moyen à une entreprise ou à un projet de mobiliser des capitaux.
  - ▶ Collecter des fonds en créant puis en vendant leurs propres jetons numériques par le biais de crowdfunding « financement participatif » sur une blockchain.
  - ▶ Les jetons peuvent être convertis dans n'importe quelle devise au taux du marché en vigueur.

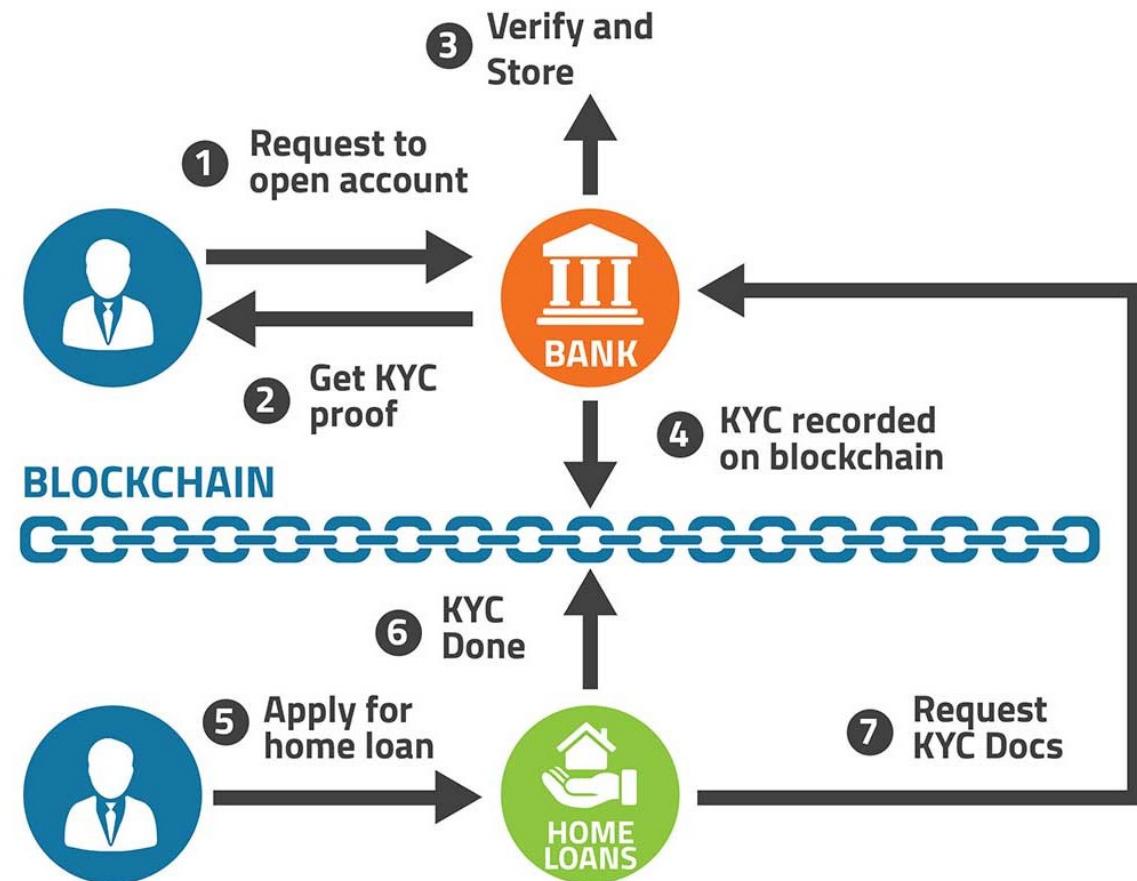
# Secteur bancaire: Know your customer (KYC)

---

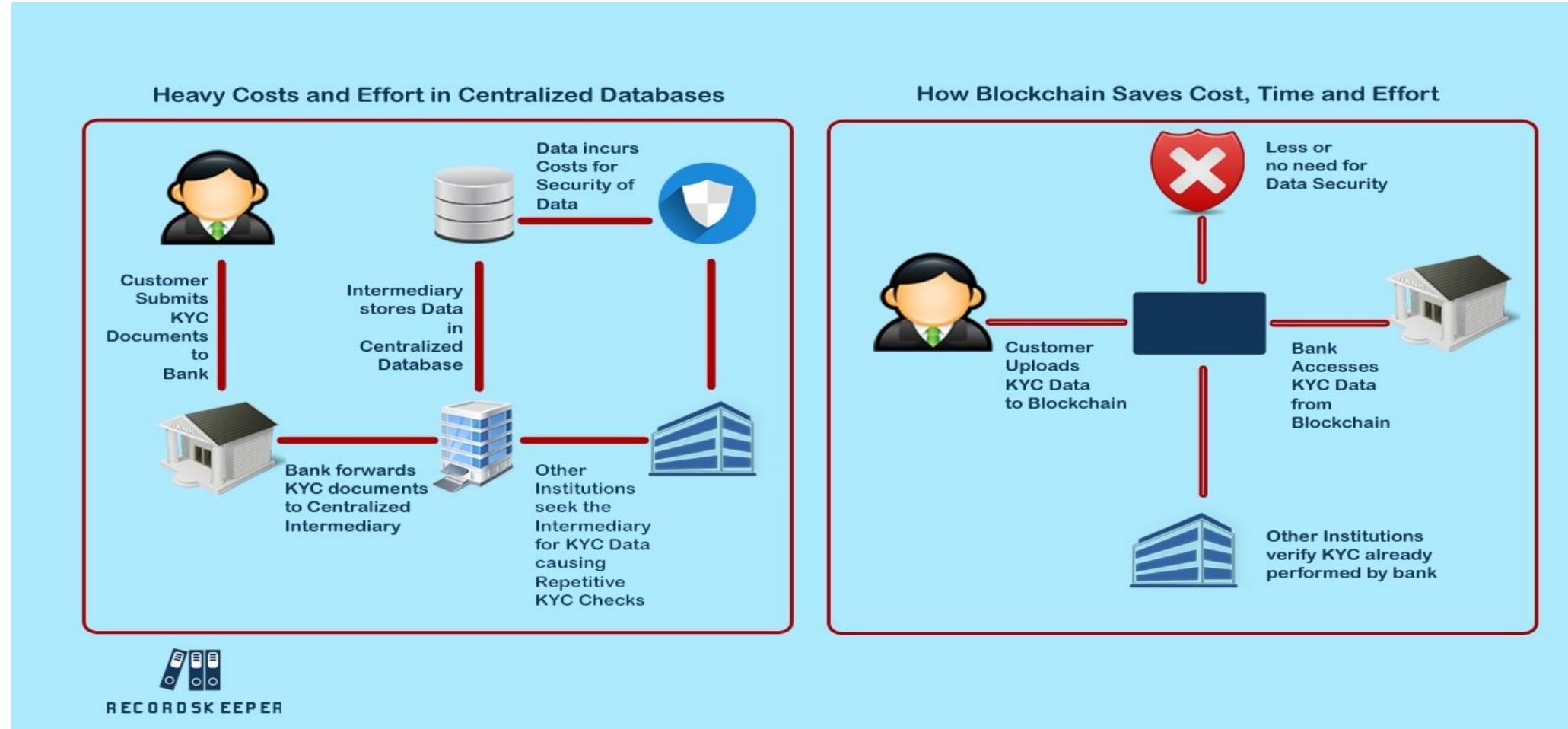
- ▶ C'est le processus qui permet de vérifier l'identité des clients d'une entreprise
  - ▶ s'assurer de la conformité des clients face aux législations anti-corruption
  - ▶ vérifier leur probité et intégrité
  - ▶ prévenir l'usurpation d'identité, la fraude fiscale, le blanchiment d'argent et le financement du terrorisme
- ▶ Ces processus se font typiquement par :
  - ▶ collecte et analyse de données,
  - ▶ vérification de la présence sur les listes (à l'exemple de celle des personnes politiquement exposées),
  - ▶ l'analyse du comportement et des transactions, etc.
- ▶ Les institutions financières consacrent une énorme quantité de ressources au processus KYC, et cela prend toujours beaucoup de temps.
- ▶ La blockchain apporte les solutions aux institutions financières

# Utilitaire KYC de Blockchain

- ▶ Les informations personnelles du client, la documentation KYC et les données sont cryptées et ajoutées sous forme de bloc dans la blockchain.
- ▶ Le blocage du client est validé en utilisant un modèle de consensus fonctionnant sur le réseau.

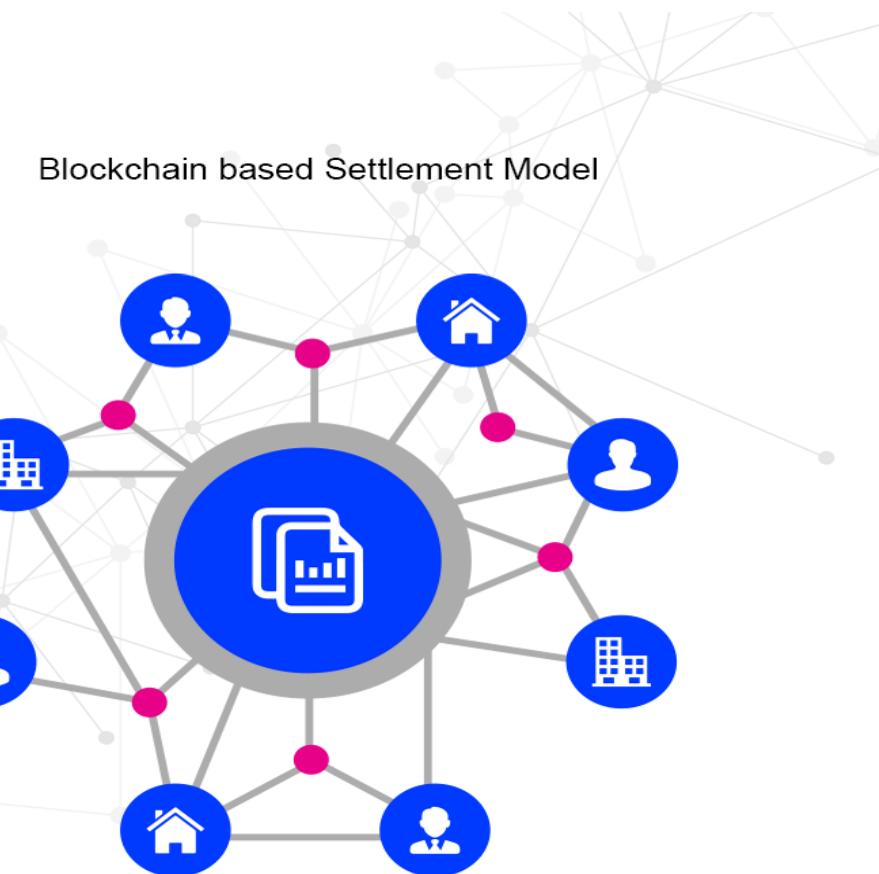
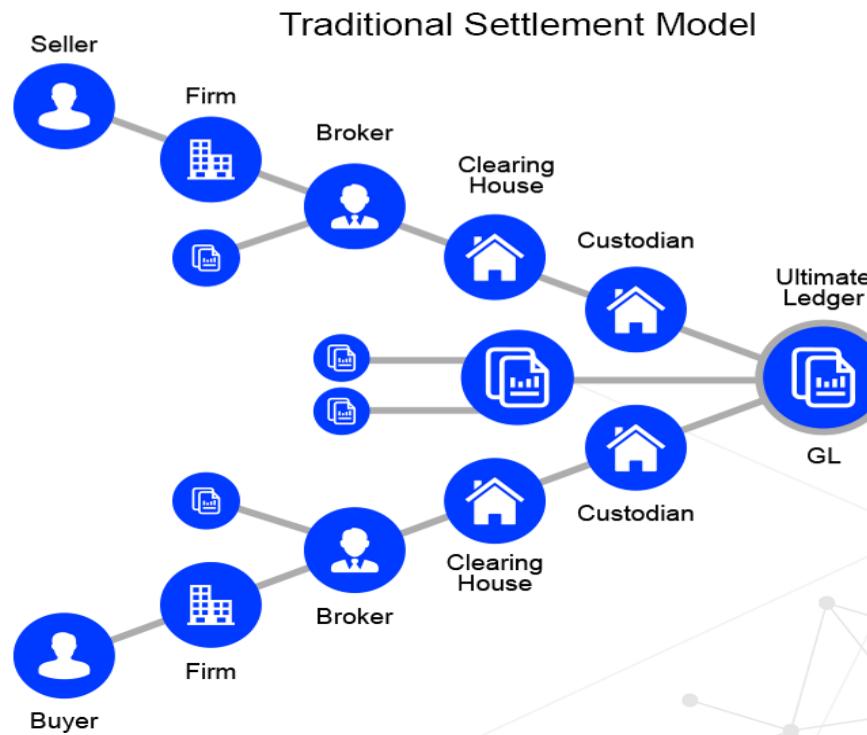


# Utilitaire KYC de Blockchain

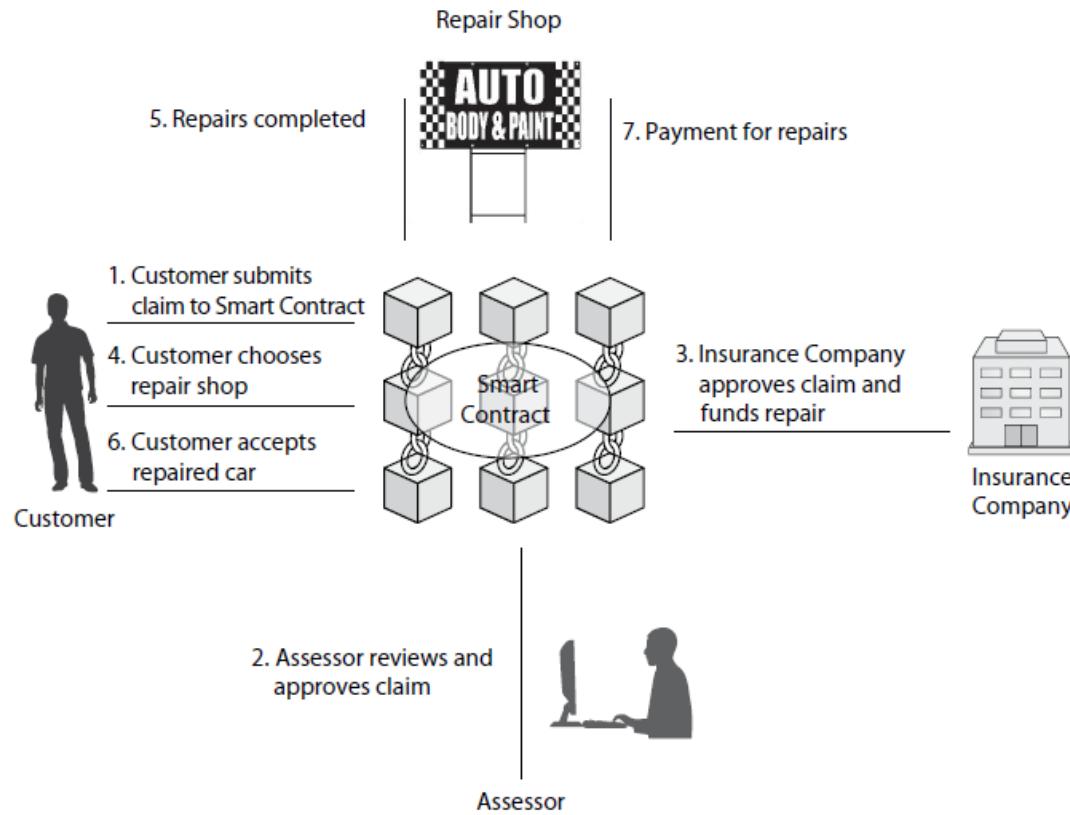


<https://openledger.info/services/blockchain-development/>

# Règlement de gestion des actifs



# Traitement des réclamations d'assurance



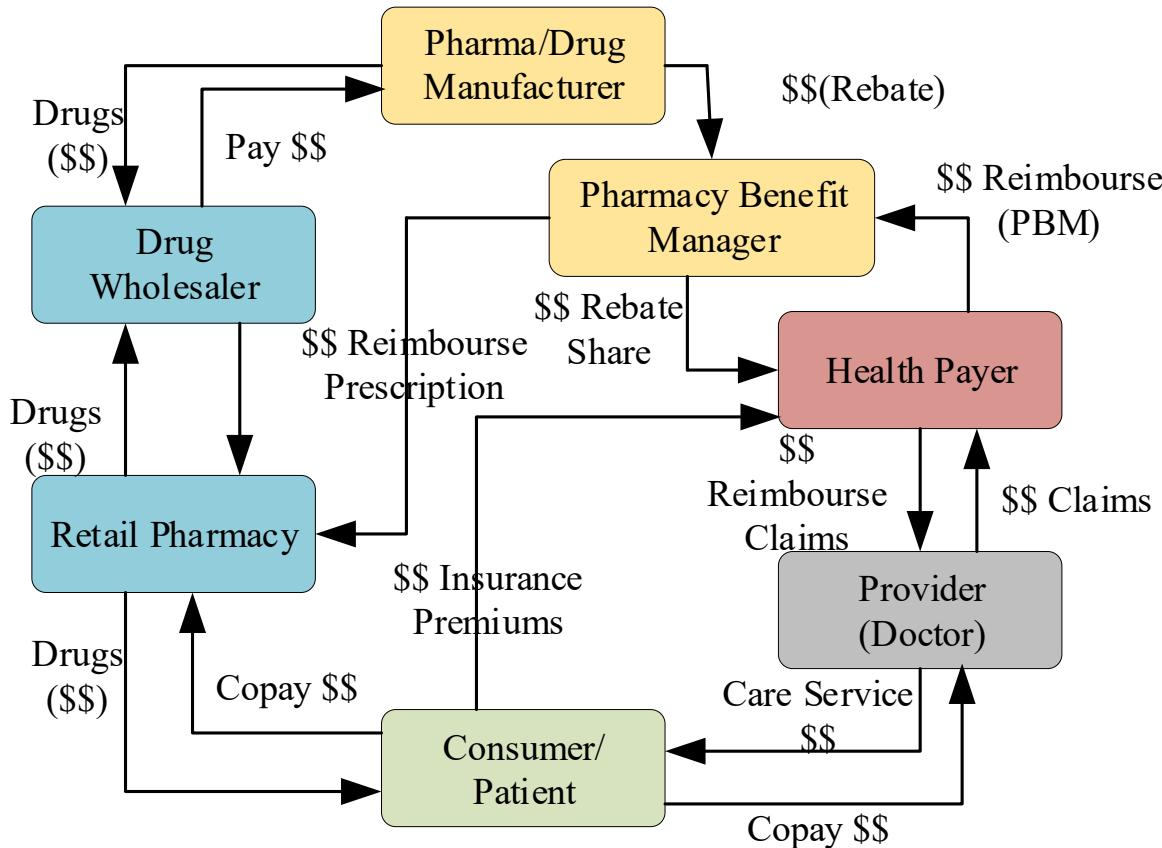
1. Le client soumet sa réclamation au Smart Contract
2. L'évaluateur examine et approuve la demande
3. Une compagnie d'assurance approuve la réclamation et la réparation des fonds
4. Le client choisit un atelier de réparation
5. Réparations effectuées
6. Le client accepte la voiture réparée
7. Paiement des réparations

<https://www.black.insure/>

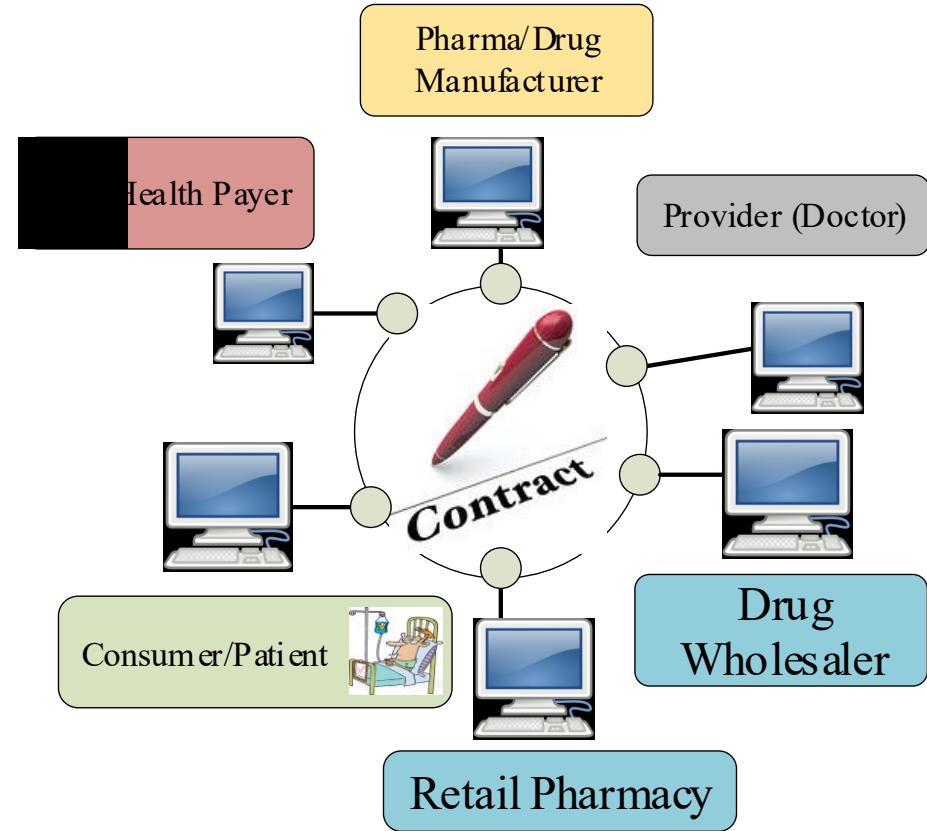
<https://b3i.tech/home.html>

# Chaîne d'approvisionnement pharmaceutique

## Modèle classique

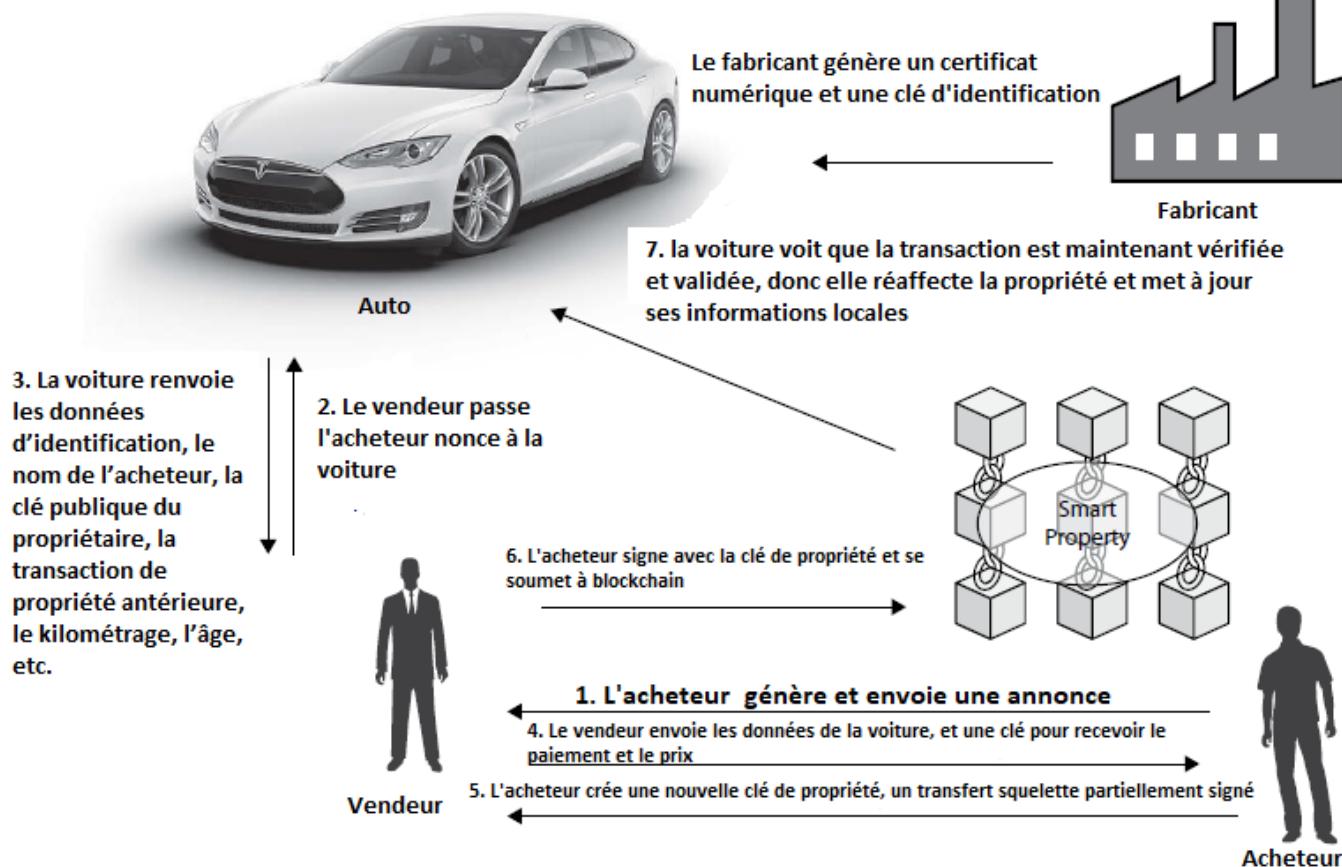


## Modèle Blockchain



# Transfert de propriété d'automobile

La technologie automobile contient des données précieuses: preuve d'existence, date de construction, historique de propriété récente, historique de maintenance, durée/distance, etc.



▶ Les voitures autonomes utilisent une algorithme de consensus pour échanger la propriété entre différents acheteurs

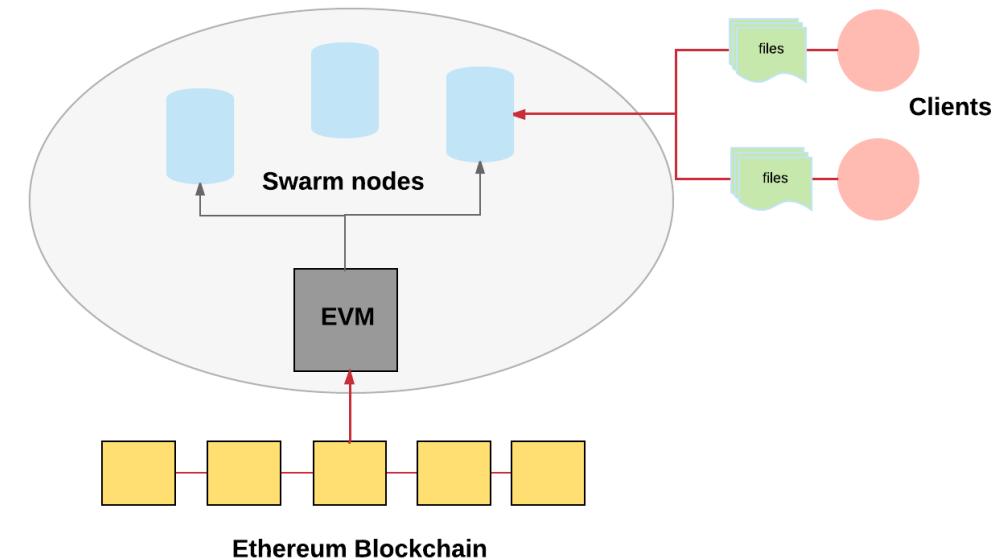
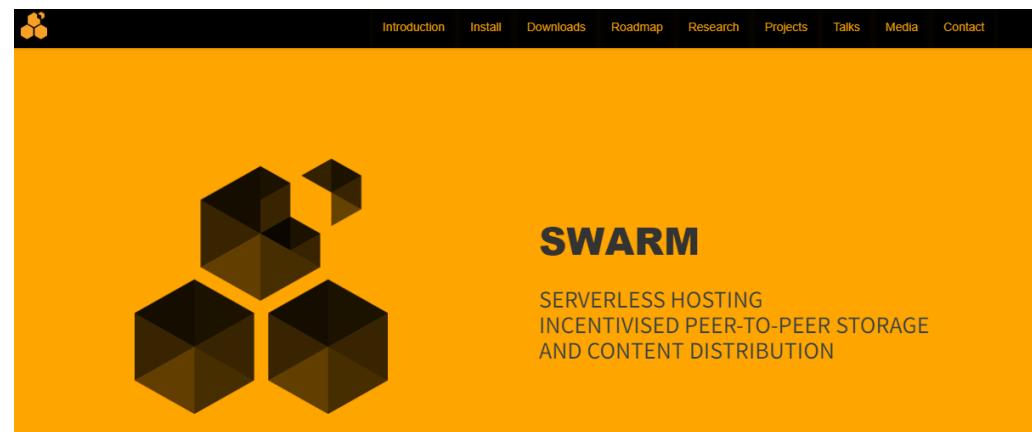
▶ PoO: Proof-of-Ownership

▶ Exemple:

- ▶ Cube ICO - Blockchain Security pour Tesla Cars
- ▶ <https://cubeint.io/>

# Système de fichiers distribués (IPFS)

- ▶ Swarm est une plate-forme de stockage distribuée et un service de distribution de contenu, un service de couche de base natif de la pile Ethereum web3.
- ▶ L'objectif premier de Swarm est de fournir un stockage décentralisé et redondant des archives publiques d'Ethereum, en particulier pour



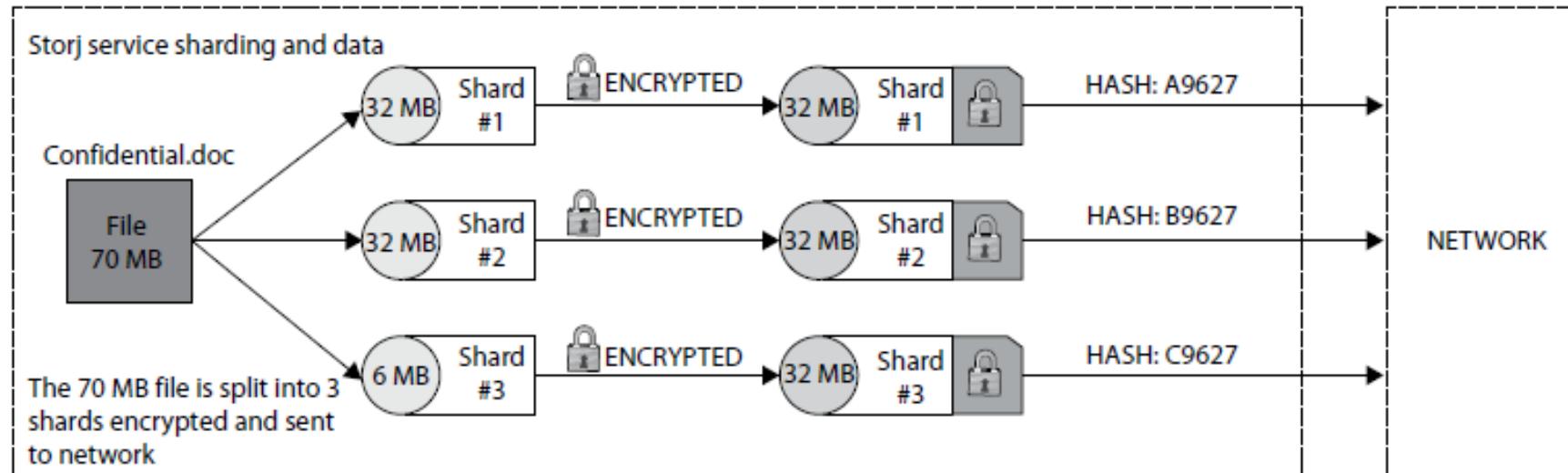
<https://memory.io/>

<https://ethersphere.github.io/swarm-home/>

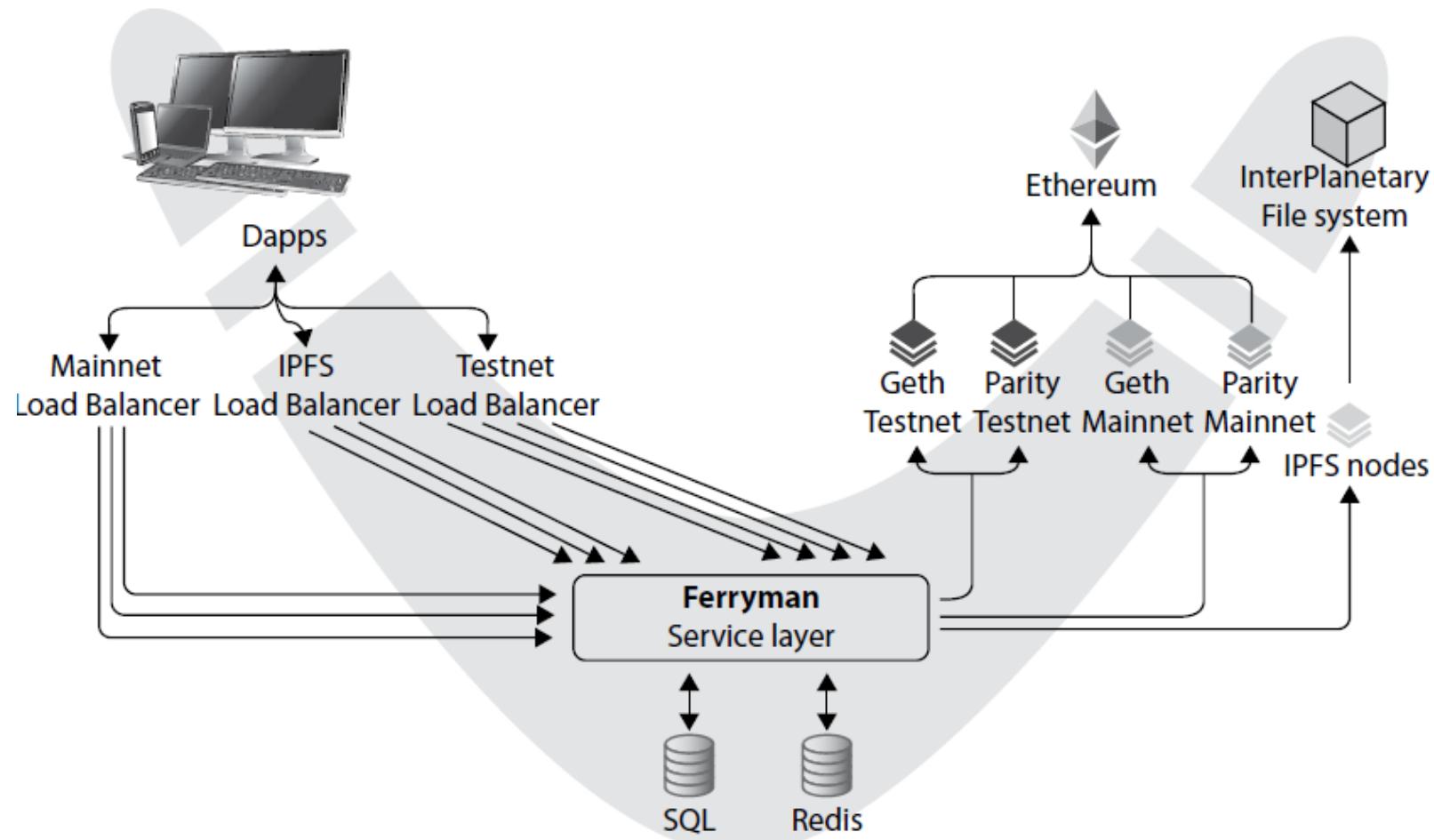
# Système de fichiers distribués (IPFS)

## ▶ Storj Storage Nodes

- ▶ Il utilise le cryptage, le partage de fichiers et une table de hachage basée sur une chaîne de blocs pour stocker les fichiers sur un réseau peer-to-peer.
- ▶ L'objectif est de rendre le stockage de fichiers dans le cloud plus rapide, moins cher et plus privé.

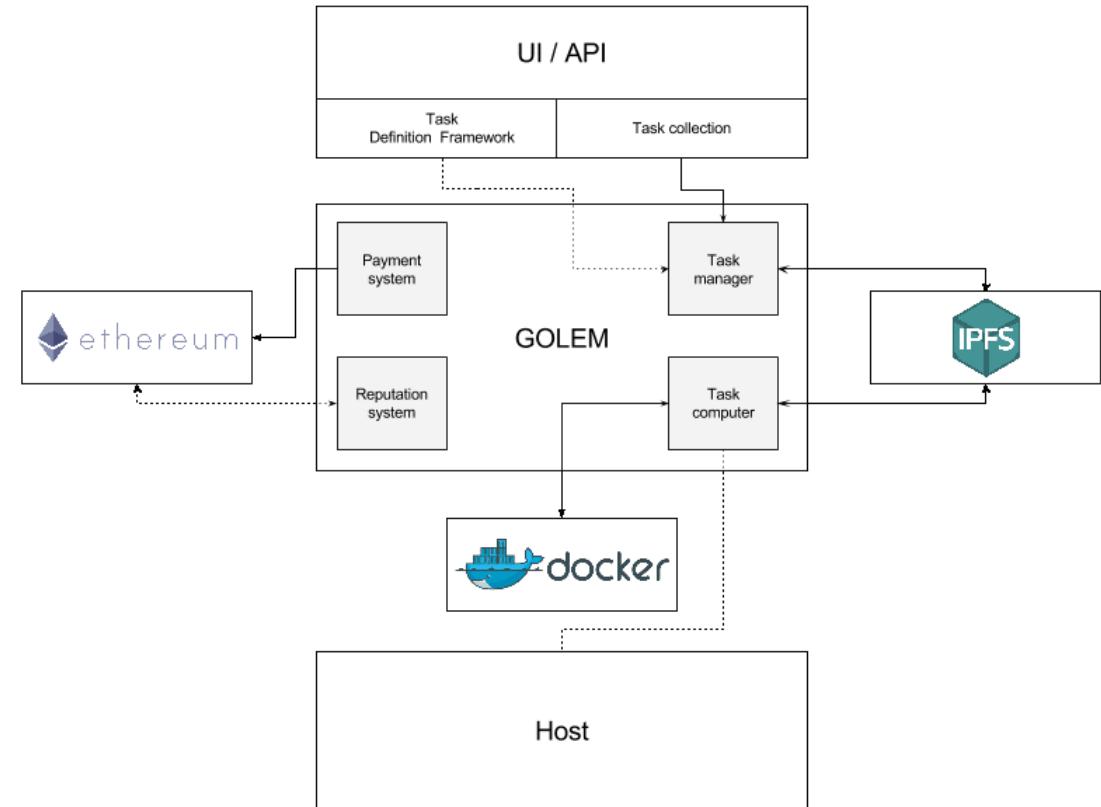


# Système de fichiers distribués (IPFS): INFURA



# Golem Super Computer

- ▶ Golem est le premier superordinateur réellement décentralisé, créant un marché mondial pour la puissance de calcul.
- ▶ Il connecte les ordinateurs au sein d'un réseau peer-to-peer,
- ▶ Il permet ainsi aux propriétaires d'applications et aux utilisateurs individuels ("demandeurs") de louer des ressources des ordinateurs des autres utilisateurs ("fournisseurs").

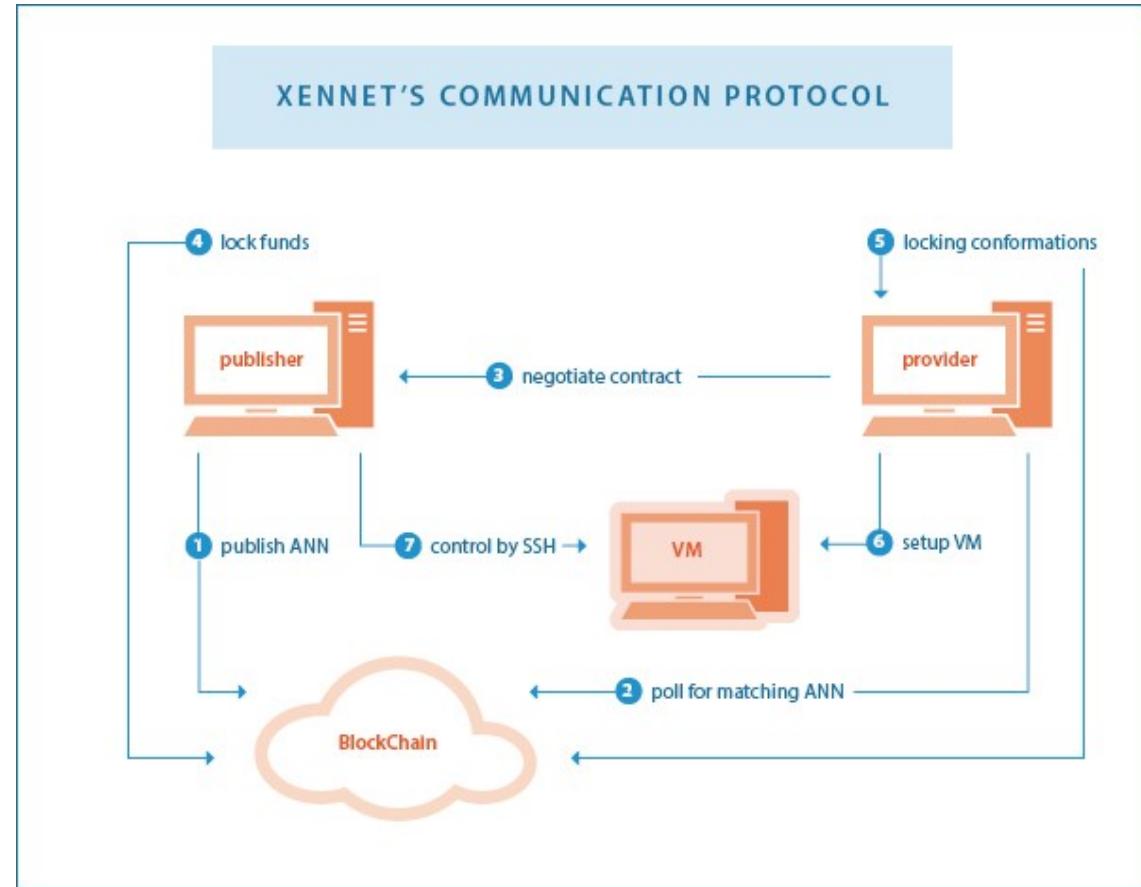


<https://golem.network/crowdfunding/Golemwhitepaper.pdf>

# ZENNET: un super calculateur public

- ▶ Zennet est un superordinateur public, distribué et décentralisé
- ▶ Offre des services de calculs distribués et virtualisation
- ▶ Par exemple, en tapant 'zennet -n 100 -m 512M', nous
  - ▶ recevant instantanément une liste de 100 ports locaux, chaque port amène à un hôte distant différent doté de 512 Mo de RAM.

<http://zennet.sc/>

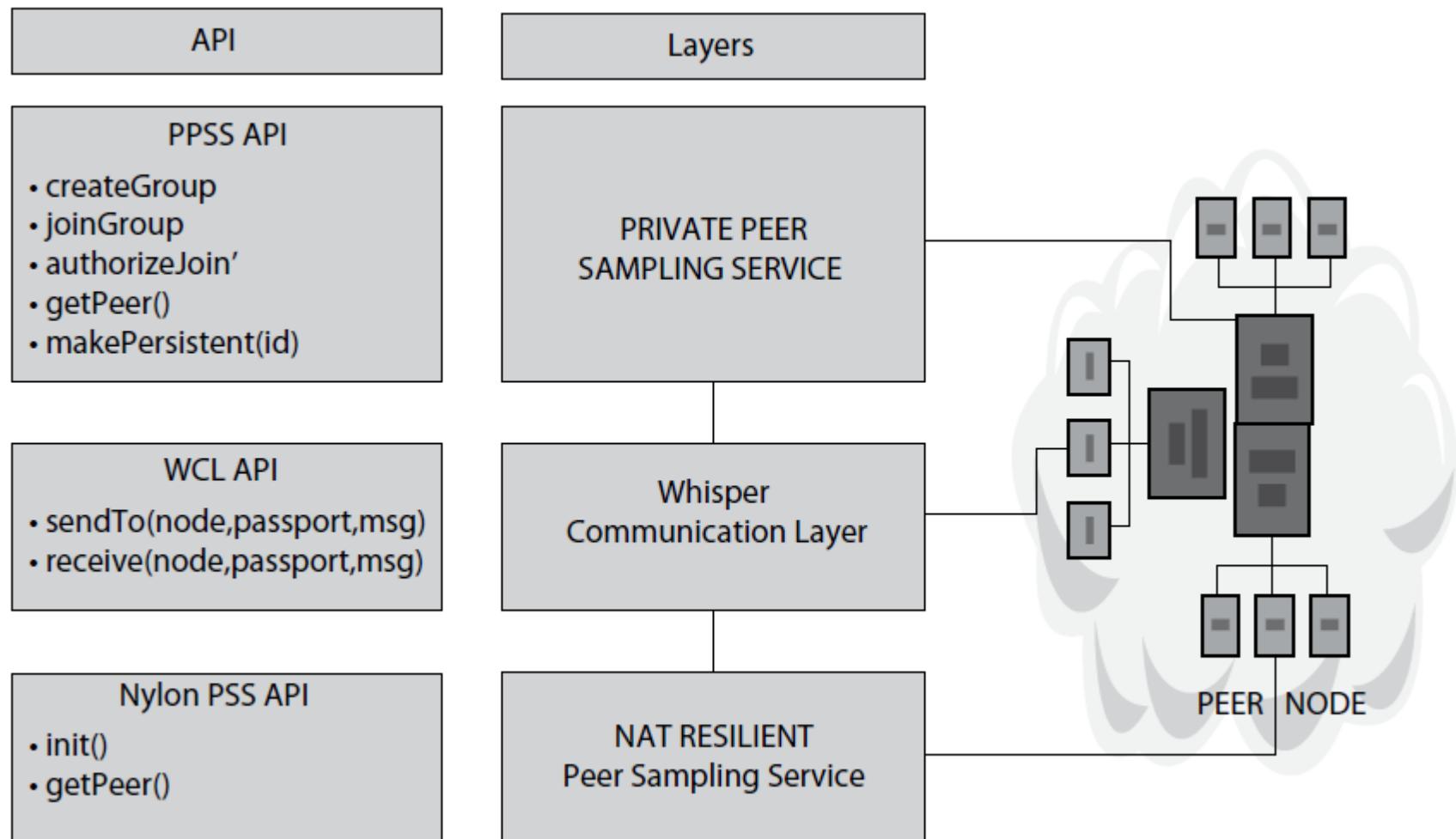


# Communications décentralisées

---

- ▶ Permet des services de communication interopérables en mode natif
- ▶ Permet d'utiliser en toute confiance des connexions peer-to-peer sans avoir à utiliser les autorités ou les services centraux (en comparaison avec VPN)
- ▶ Exemples d'applications:
  - ▶ Systèmes de partage d'informations garantissant la liberté d'expression
  - ▶ Applications tchat décentralisées:  
<https://blockchainwhispers.com/#>
- ▶ Whisper est un protocole de communication permettant aux DApp de communiquer entre eux
- ▶ C'est un middleware développé pour la blockchain Ethereum pour gérer des communications décentralisées entre Dapps
- ▶ La prochaine version sera standardisée par le Web3 Foundation
- ▶ <https://github.com/w3f/messaging/>

# Couches d'architecture Whisper



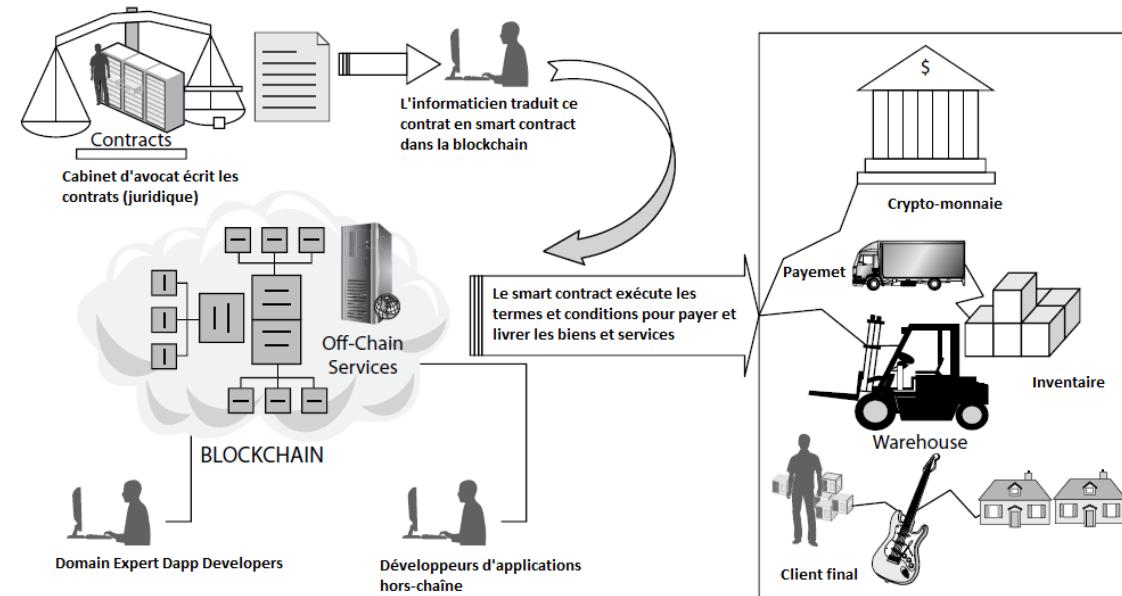
# Législation sur la blockchain et la vie privée

- ▶ Votre passeport dans la blockchain
  - ▶ C'est un projet incubateur open source pour permettre aux gouvernements de leurs propres services de passeport privé
  - ▶ Il peut être utilisé pour valider et prouver l'existence d'autres personnes en utilisant uniquement les outils de la blockchain.
  - ▶ ObjectTech fournit un passeport numérisé qui intègre une fonctionnalité d'identité autonome
  - ▶ Le Blockchain passeport permet de protéger la vie privée des passagers, en contrôlant les données et les parties qui consultent le passeport.



# Blockchain et Aspect juridique du contrat autonome

- ▶ Blockchain pour le droit d'auteurs
  - ▶ Une œuvre peut avoir une signature électronique dans la blockchain, conservée de façon immutable
  - ▶ La blockchain joue un rôle de pare-feu contre la reproduction illicite
- ▶ Les smart contracts ont une valeur juridique, de la même manière que les contrats traditionnels (écrit sur papier)



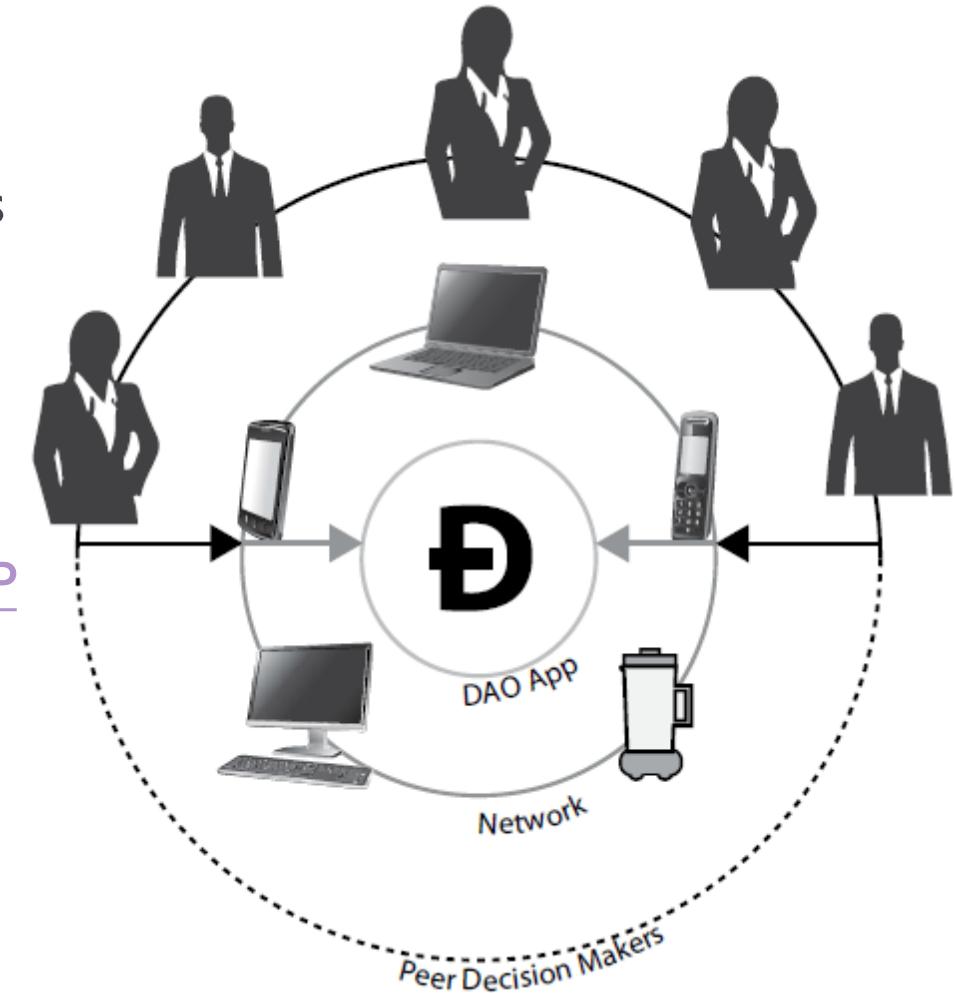
# Registre de commerce dans la blockchain

---

- ▶ L'état de Delaware a créé une blockchain pour gérer les identités des entreprises de manière décentralisées
  - ▶ Un registre de commerce distribué entre les différentes entités intervenantes
- ▶ IBM et Proxeus permettent d'enregistrer une nouvelle entreprise dans le registre de commerce en 3 heures (au lieu de 3-4 semaines)
  - ▶ en utilisant des workflows numérisés et des contrats intelligents.

# Decentralized Autonomous Organizations (DAO)

- ▶ Organisation qui utilise des règles codées sous forme de contrats intelligents
  - ▶ L'enregistrement des transactions financières et les règles du programme d'un DAO sont conservés dans une blockchain.
  - ▶ Le concept introduit plusieurs modèles décrits dans un « papier blanc »  
<https://download.slock.it/public/DAO/WhitePaper.pdf>



# Decentralized Autonomous Organizations (DAO)

---

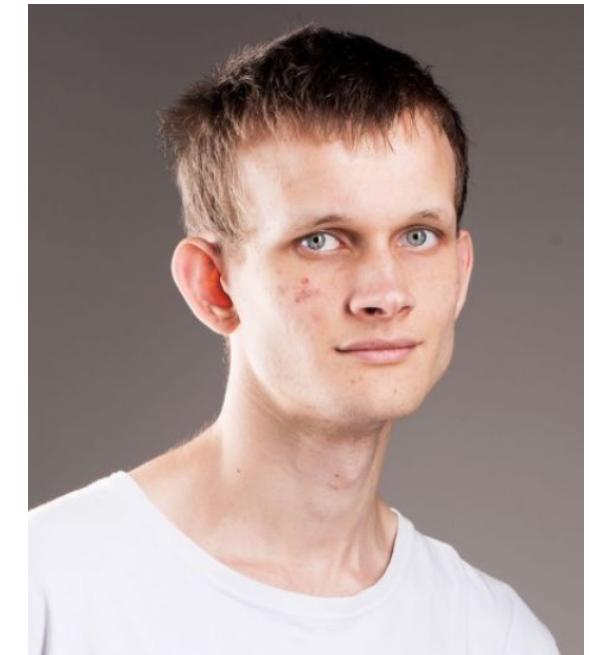
- ▶ Les DAO rendent les concepts traditionnels de propriété et de responsabilité obsolètes
  - ▶ Un DAO peut dépasser les frontières juridictionnelles car les nœuds d'une blockchain peuvent être situés dans n'importe quel pays du monde.
  - ▶ Cela pose des problèmes juridictionnels complexes qui nécessiteront un examen spécialisé de la part des avocats connaissant parfaitement le droit de chaque pays et les relations contractuelles correspondantes.
- ▶ Exemples de DAOs:
  - ▶ DIGIX: <https://digix.global/dgd/>
  - ▶ MakerDAO <https://makerdao.com/en/>



# Introduction à Ethereum

# Introduction d'Ethereum

- ▶ Vitalik Buterin, un jeune programmeur de Bitcoin agé de 20 ans, commence à penser à une alternative à Mastercoin (protocole de base de Bitcoin) <https://github.com/mastercoin-MSC>
- ▶ Il propose en 2013 une approche généralisé plus flexible de Mastercoin, et introduit des smart contrats scriptable
- ▶ Les développeurs de Bitcoin sont impressionnés par cette solution, et changent leur feuille de route de développement de Bitcoin
- ▶ En 2014, une version initiale d'Ethereum est introduite et devient alors la technologie Blockchain la plus prometteuse
- ▶ En mars 2017, la **Ethereum Enterprise Alliance** a été formée. La liste des organisations participantes comprend Microsoft, Intel, J.P. Morgan, BNY Mellon, etc.
- ▶ <https://entethalliance.org/members/>



# Ethereum

---

- ▶ Ethereum est une infrastructure informatique open source et globalement décentralisée qui exécute des programmes appelés **smart contrats**
- ▶ Les smart contrats sont des applications informatique qui fonctionnent exactement comme elles ont été programmées (haute disponibilité, pas de censure, ni fraude ou d'interférence de tiers).
- ▶ Les applications Ethereum s'exécutent sur une blockchain personnalisée, une infrastructure globale partagée pouvant déplacer la valeur et représenter la propriété « Ownership ».
- ▶ Ethereum utilise une blockchain pour synchroniser et stocker les changements d'état du système, ainsi qu'une crypto-monnaie appelée Ether pour mesurer et limiter les coûts des ressources d'exécution.
- ▶ **Au cœur d'Ethereum : les applications décentralisées («Dapps »)**

# Ethereum vs Bitcoin

---

- ▶ Ethereum partage de nombreux éléments avec d'autres blockchains ouverts:
  - ▶ un réseau peer-to-peer connectant des participants,
  - ▶ un algorithme de consensus byzantin à tolérance de pannes pour la synchronisation des mises à jour d'état
  - ▶ utilisation de primitives cryptographiques telles que la signature numérique, hachages, et une crypto monnaie numérique (Ether).
- ▶ Mais, elle diffère
  - ▶ Ethereum n'a pas été développée pour être un réseau de paiement en crypto monnaie, mais Ether est conçu pour financer l'utilisation de la plateforme Ethereum comme un super-ordinateur mondial,
  - ▶ Contrairement à Bitcoin qui utilise un langage script limité et est intentionnellement limité à une simple évaluation vrai/faux des conditions de dépense, Ethereum est conçue pour être une blockchain programmable à usage général.
  - ▶ Ethereum est « Turing complete » et peut être utilisé comme un ordinateur ordinaire

# Développement de DApps avec Ethereum

---

- ▶ Ethereum a été développé pour être une blockchain polyvalente pour développer de diverse applications,
- ▶ Mais rapidement, la vision d'Ethereum s'est élargie pour devenir une plateforme de programmation DApps.
- ▶ Une application DApps a besoin
  - ▶ un contrat intelligent (smart contract)
  - ▶ une interface utilisateur Web (frontend).
- ▶ une DApp est une application Web construite sur des services d'infrastructure ouverts, décentralisés et d'égal à égal.

# Exemple d'Ethereum DApps

---

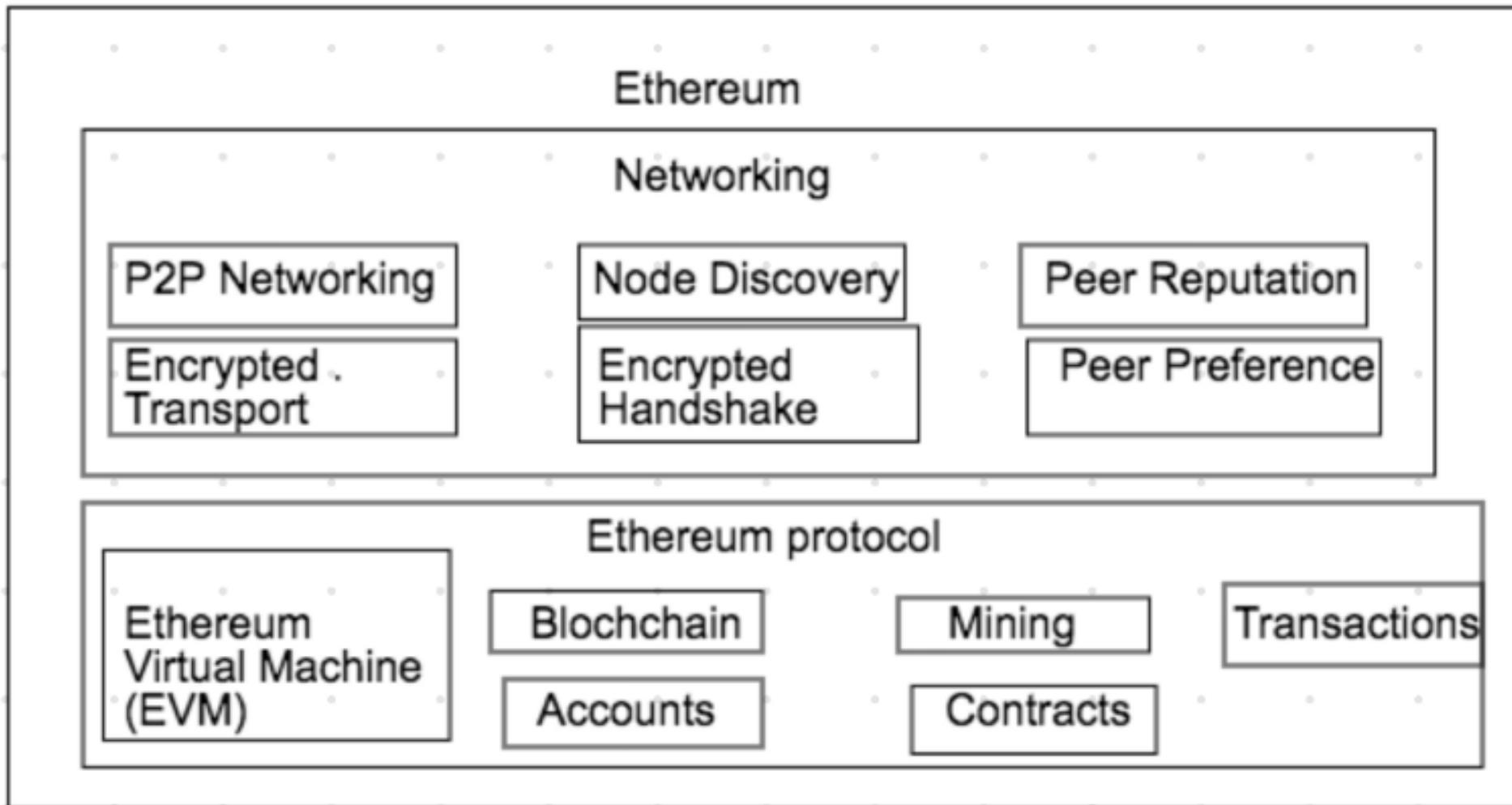
- ▶ Diverses applications décentralisées(Dapps) peuvent être développées:
  - ▶ [UjoMusic](#) veut permettre aux artistes de placer leurs titres sur une blockchain, et de récolter ensuite les droits d'auteur automatiquement, grâce à des smart contracts.
  - ▶ [Augur](#) propose une plateforme décentralisée de marché prédictif basée sur la blockchain, utilisant là encore des smart contracts.
  - ▶ [Transactive Grid](#) veut permettre à chacun de vendre et d'acheter des crédits d'énergie à son voisin, de façon pair-à-pair.
  - ▶ [Filecoin](#) est un service de stockage cloud décentralisé, qui permet de louer à d'autres utilisateurs l'espace libre de son ordinateur,
  - ▶ [Golem](#) permettra à chacun de louer la puissance de calcul inutilisée sur ses appareils (ordinateur, smartphone...) pour tout type d'applications (analyses Big Data en recherche médicale, etc.).

# Ethereum Cryptomonnaie

- ▶ L'unité monétaire d'Ethereum est appelée éther, également identifiée par «ETH» ou par les symboles « E »
- ▶ L'éther est subdivisé en unités plus petites, jusqu'à la plus petite unité possible, appelée « wei » : 1 ETH = 1\*10<sup>18</sup> wei
- ▶ <https://etherconverter.online/>

Value (in wei)	Exponent	Common name	SI name
1	1	wei	Wei
1,000	10 <sup>3</sup>	Babbage	Kilowei or femtoether
1,000,000	10 <sup>6</sup>	Lovelace	Megawei or picoether
1,000,000,000	10 <sup>9</sup>	Shannon	Gigawei or nanoether
1,000,000,000,000	10 <sup>12</sup>	Szabo	Microether or micro
1,000,000,000,000,000	10 <sup>15</sup>	Finney	Milliether or milli
1,000,000,000,000,000,000	10 <sup>18</sup>	Ether	Ether
1,000,000,000,000,000,000,000	10 <sup>21</sup>	Grand	Kiloether
1,000,000,000,000,000,000,000,000	10 <sup>24</sup>		Megaether

# Ethereum Blockchain Stack



# Portefeuille de Ethereum

---

- ▶ La portefeuille « Wallet » est une application logicielle pour gérer un compte Ethereum, c'est une « passerelle » entre l'utilisateur et Ethereum
- ▶ Il existe une variété de portefeuilles (différentes caractéristiques et designs)
  - ▶ **MetaMask** est un portefeuille d'extensions de navigateur qui s'exécute dans votre navigateur (Chrome, Firefox, etc.) qui peut se connecter à une variété de nœuds Ethereum et de blockchains de test <https://metamask.io/>
  - ▶ **Jaxx** est un portefeuille multiplateforme et multidevises fonctionnant sur divers systèmes d'exploitation (Android, iOS, Windows, macOS, and Linux). <https://jaxx.io/>
  - ▶ **MyEtherWallet (MEW)** est un portefeuille basé sur le Web qui fonctionne dans n'importe quel navigateur. <https://www.myetherwallet.com/>
- ▶ Les portefeuilles sont utiliser **pour contrôler et gérer les clés privés** du compte Blockchain

# Porte monnaie de Ethereum (Hardware)

- ▶ Il y a aussi des Wallets sous forme de matériel:

- ▶ Ledger Nano X <https://www.ledger.com/>
- ▶ TREZOR One <https://trezor.io/>
- ▶ KeepKey <https://shapeshift.io/keepkey/>
- ▶ Archos Safe-T mini [https://www.archos.com/products/crypto/archos\\_safetmini/index.html](https://www.archos.com/products/crypto/archos_safetmini/index.html)
- ▶ <https://bitcoin.org/fr/wallets/hardware/>

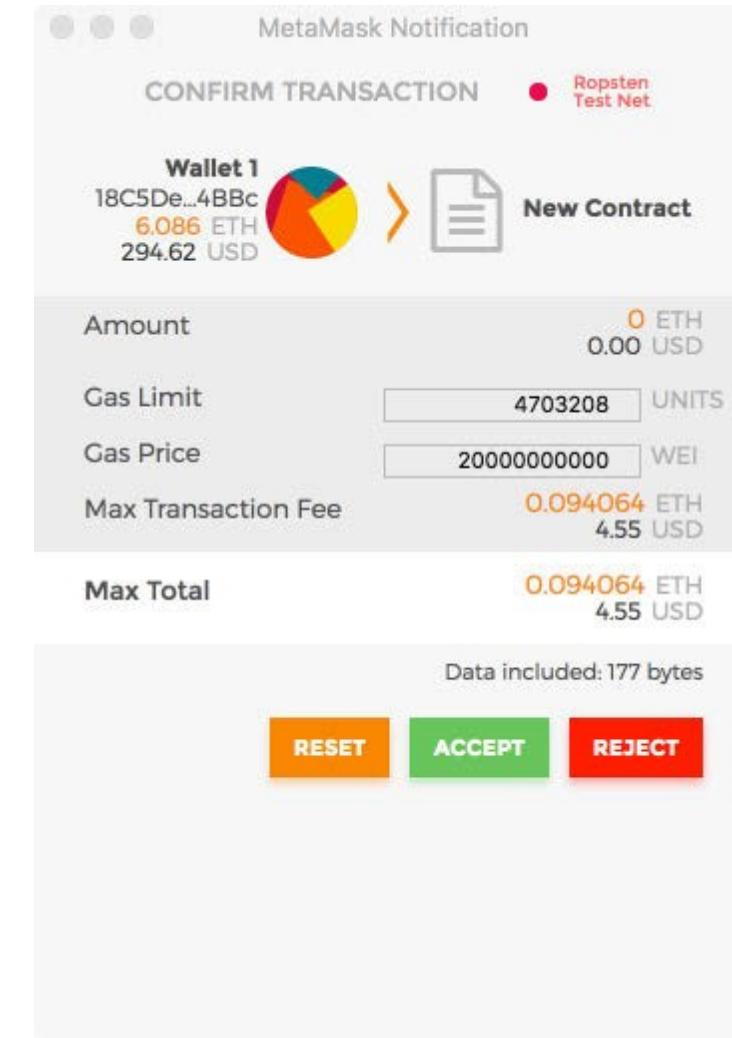
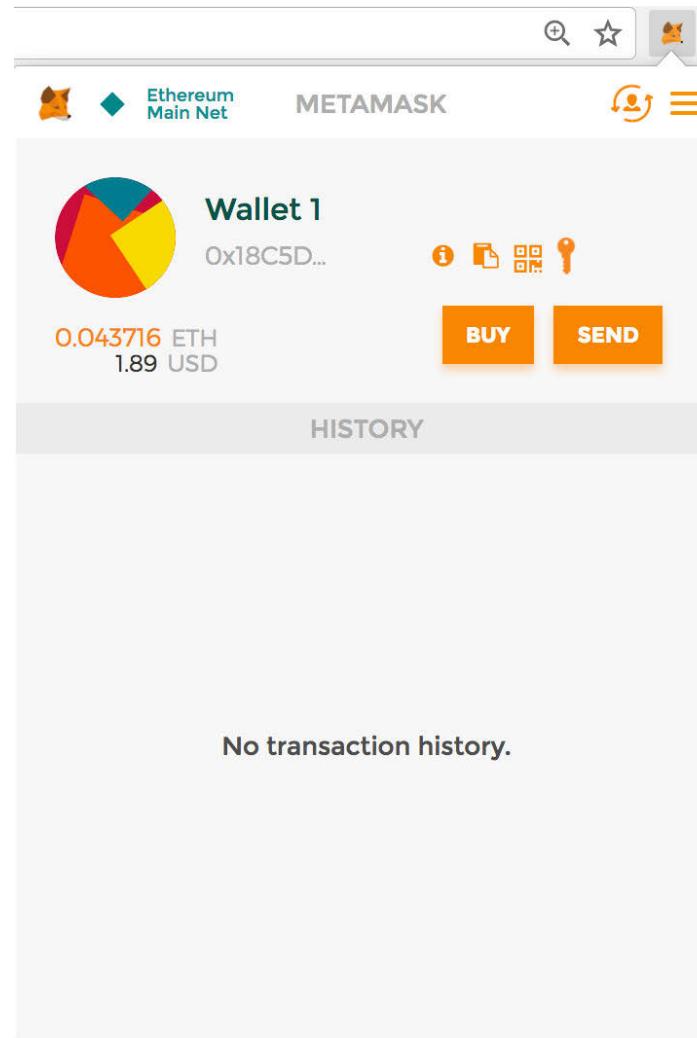


# Contrôle et gestion du compte

---

- ▶ Le contrôle de compte Blockchain devient une grande responsabilité
  - ▶ Si on perd la clé privée, on ne peut plus accéder à ses fonds et smart contrats
  - ▶ Personne ne peut vous aider à retrouver l'accès - vos fonds seront bloqués pour toujours.
  - ▶ Une tragédie ?
    - ▶ €125M manquants après la mort du fondateur du crypto-exchange QuadrigaCX
    - ▶ <https://cryptonaute.fr/e125m-manquants-mort-fondateur-crypto-exchange-quadriga-cx/>
  - ▶ Il faut prendre des mesures de haute sécurité
    - ▶ Ne jamais enregistrer votre clé sous forme de texte plain (numérique), utiliser un logiciel, ou bien hardware Wallet
    - ▶ Les clés privées peuvent être stockées sous une forme cryptée
    - ▶ Avant de transférer des montants importants (en particulier vers de nouvelles adresses), effectuez d'abord une petite transaction test (par exemple, une valeur inférieure à 1 \$) et attendez la confirmation de réception.

# Exemple de portefeuille: MetaMask <https://metamask.io>



# Réseaux de Blockchain Ethereum

---

- ▶ Ethereum utilise 04 types de réseaux Blockchain (MainNet et TestNet)
  - ▶ Main Ethereum Network
    - ▶ La principale blockchain publique Ethereum.
    - ▶ ETH réel, valeur réelle et conséquences réelles.
  - ▶ Ropsten Test Network
    - ▶ C'est une Blockchain Ethereum publique de test dans le réseau.
    - ▶ La cryptomonnaie ETH sur ce réseau n'a aucune valeur.
  - ▶ Kovan Test Network
    - ▶ Blockchain Ethereum publique de test qui utilise un protocole de consensus Aura avec preuve d'autorité (proof-of-authority)
    - ▶ Le réseau de test Kovan est pris en charge par le client Ethereum Parity uniquement.
    - ▶ La cryptomonnaie ETH sur ce réseau n'a aucune valeur.
  - ▶ Rinkeby Test Network
    - ▶ Blockchain publique de test Ethereum avec un protocole de consensus « Clique »

# Blockchain sur votre PC portable

---

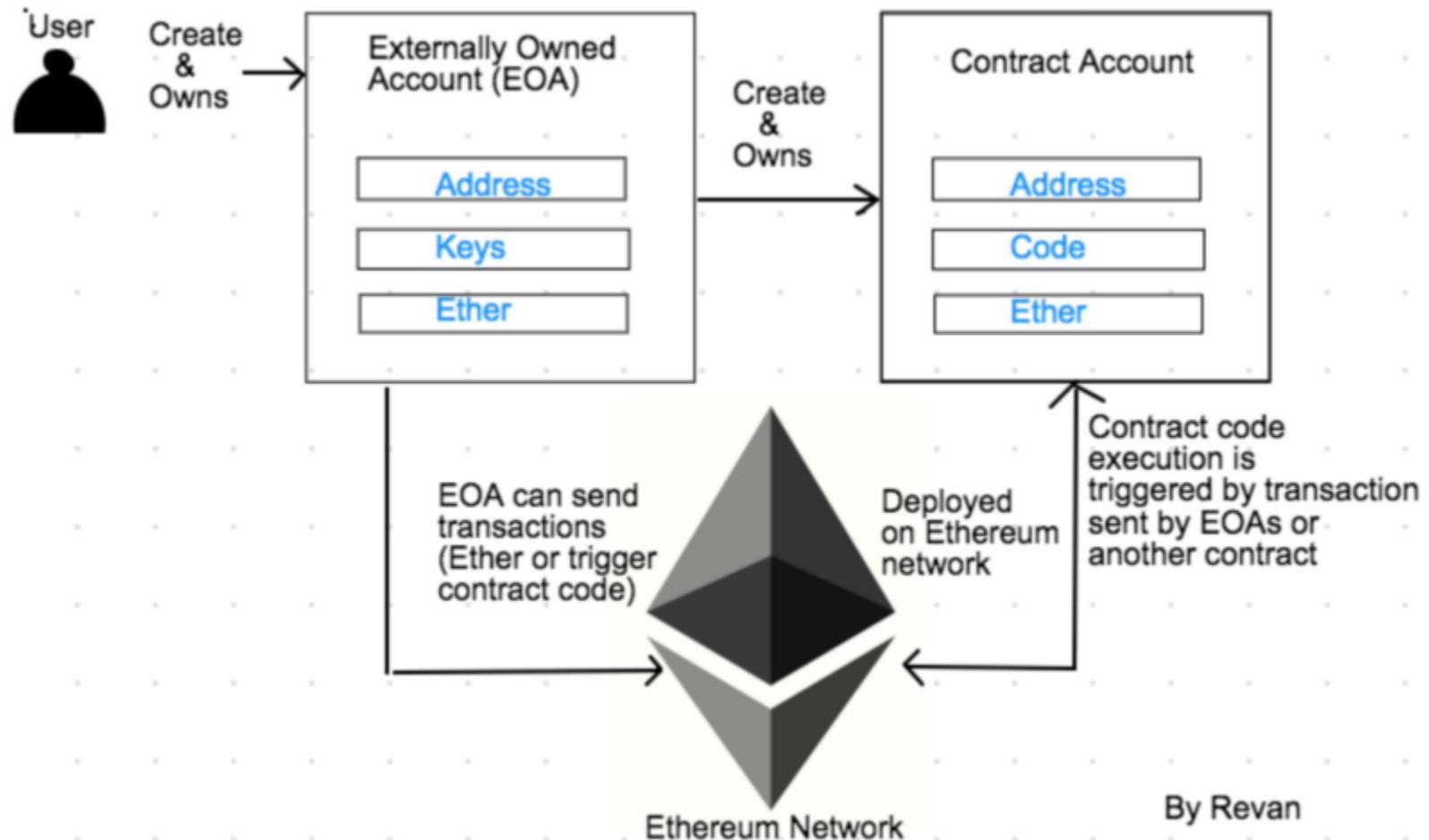
- ▶ **Localhost 8545**
  - ▶ Se connecte à un noeud s'exécutant sur le même ordinateur que le navigateur.
  - ▶ Le nœud peut faire partie d'une blockchain publique (main ou testnet) ou d'un réseau de test privé.
- ▶ **Custom RPC**
  - ▶ Vous permet de connecter MetaMask à n'importe quel nœud doté d'une interface RPC (Remote Procedure Call) compatible avec le client Ethereum Geth.
  - ▶ Le nœud peut faire partie de toute blockchain publique ou privée.

# Exemple de Démonstration

---

- ▶ **Etape 1:**
  - ▶ Installer l'extension MetaMask dans votre navigateur
  - ▶ Créer un compte et connecter vous à Ethereum Blockchain
  - ▶ Vous allez remarquer l'ensemble de réseaux Blockchain de Ethereum (MainNet et TestNets)
  - ▶ Choisir un réseau de Test Ethereum, i.e. Ethereum Ropsten, pour utiliser de la fausse cryptomonnaie Eth
- ▶ **Etape 2:**
  - ▶ Connecter vous à <https://faucet.metamask.io/> pour recharger votre compte avec Eth
  - ▶ Envoyer un Ether à votre compte (vous pouvez envoyer 1ETH à chaque fois)
  - ▶ Regarder la transaction (ID) générée en ligne de votre achat de la fausse ETH
  - ▶ Maintenant, faites l'inverse en envoyer 1 ETH depuis votre compte au compte FAUCET
  - ▶ Suivre la transaction (ID) en ligne dans votre navigateur sur Ropsten blockchain

# Type de comptes dans Ethereum



# Type de comptes dans Ethereum

---

- ▶ On distingue deux types d'adresses dans Ethereum
  - ▶ Adresse de comptes de propriété externe (Externally Owned Account (EOA)).
    - ▶ Le type de compte que vous avez créé dans le portefeuille MetaMask (Account 1, Compte 2)
    - ▶ Les comptes EOA sont ceux qui ont une clé privée pour contrôler l'accès aux fonds ou aux contrats.
  - ▶ Adresse du « smart contrat »
    - ▶ Un compte de smart contrat diffère du compte EOA.
    - ▶ un compte de contrat n'a pas de clé privée.
    - ▶ Au lieu de cela, il est détenu (et contrôlé) par la logique de son code du smart
    - ▶ Les contrats peuvent également envoyer et recevoir de l'éther, tout comme les EOA.
    - ▶ il ne peut pas initier une transaction, seuls les EOA peuvent le faire
    - ▶ les contrats peuvent réagir à ces transactions en appelant d'autres contrats, créant ainsi des chemins d'exécution complexes.

# Clés et adresses

---

- ▶ Ethereum a deux types de comptes différents:
  - ▶ les comptes détenus en externe (EOA)
  - ▶ les contrats.
- ▶ La propriété de la cryptomonnaie ETER par les EOA est établie au moyen de clés privées numériques, d'adresses Ethereum et de signatures numériques.
- ▶ Les clés privées sont au cœur de toutes les interactions de l'utilisateur avec Ethereum.
  - ▶ Les adresses de compte sont directement dérivées de clés privées: une clé privée détermine de manière unique une seule adresse Ethereum, également appelée compte.

## Successful miner's computer

Takes the following steps and broadcasts block header,  $H$ , to network

### Determine Transactions

Miner picks transactions to process (from those broadcasted)

### Determine Ommers

Miner finds and includes valid ommers

### Apply Rewards

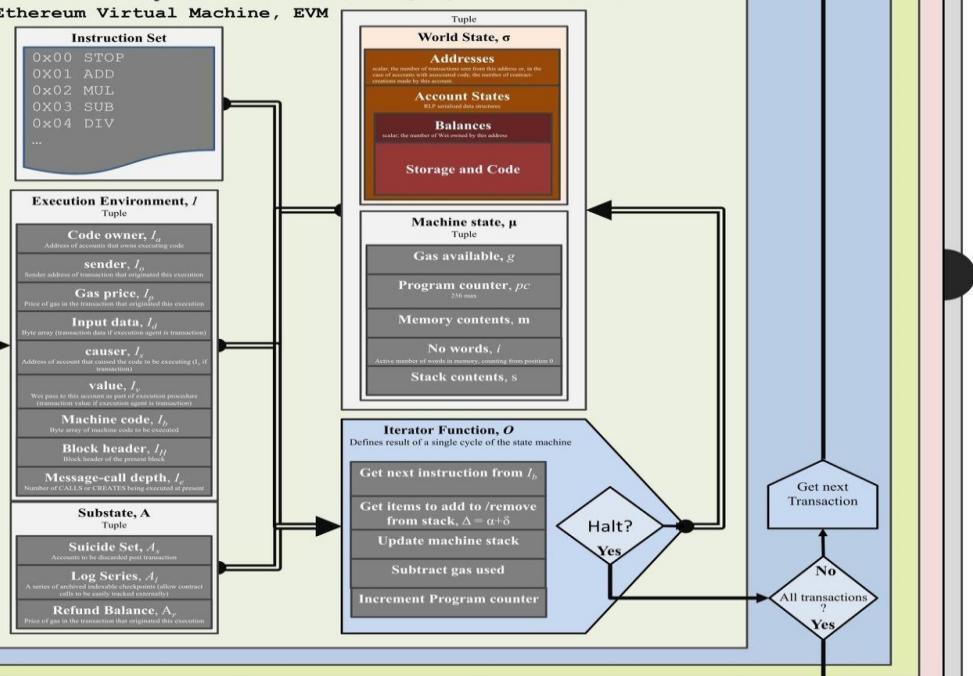
Update account balance(s) to reward valid blocks

### Compute a Valid State

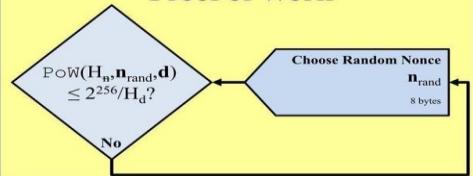
**Block Finalisation** Defines result of all selected state transitions

**State Transition Cycle** Defines result of a single transaction  $\sigma_{t+1} = Y(\sigma_t, T)$

**Execution Cycle** Defines result of a single cycle of the state machine

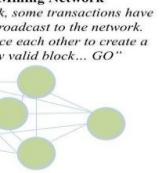


### Proof of Work

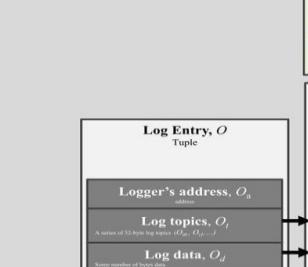
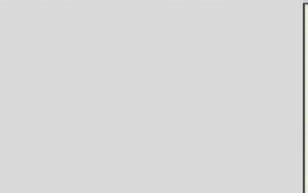
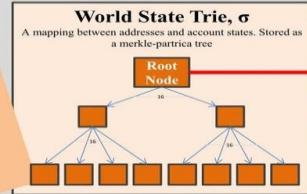
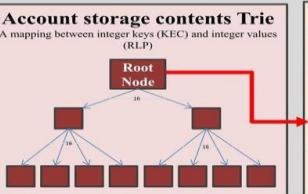


## Mining Network

"Oh look, some transactions have been broadcast to the network. Let's race each other to create a new valid block... GO!"



## Information required to derive Block Header



## Ethereum Blockchain Mechanism (Proof Of Work)

An interpretation of the Ethereum Project Yellow Paper  
G. Wood, "Ethereum: A secure decentralised generalised transaction ledger", 2014.

Lee Thomas  
2014-06-21  
Ver 0.1 2016-06-22

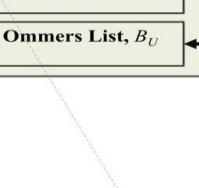
### Ethereum Network

"Oh look another miner has been successful. Therefore there is little point in continuing with this block – we'll start on the next one!"  
"note – we had to run all the same code on our copies of the EVM to prove this!"

### Block, B



### Transaction List, B\_T



### Ommers List, B\_U



## Introduction à la programmation avec Solidity



# Solidity

 Solidity  
latest

Search docs

---

- Introduction to Smart Contracts
- Installing the Solidity Compiler
- Solidity by Example
- Solidity in Depth
- Security Considerations
- Resources
- Using the compiler
- Contract Metadata
- Contract ABI Specification
- Yul
- Style Guide
- Common Patterns
- List of Known Bugs
- Contributing
- LLL

---

Docs » Solidity

 Edit on GitHub

## Solidity

<https://solidity.readthedocs.io/en/latest/>



Solidity is an object-oriented, high-level language for implementing smart contracts. Smart contracts are programs which govern the behaviour of accounts within the Ethereum state.

Solidity was influenced by C++, Python and JavaScript and is designed to target the Ethereum Virtual Machine (EVM).

Solidity is statically typed, supports inheritance, libraries and complex user-defined types among other features.

With Solidity you can create contracts for uses such as voting, crowdfunding, blind auctions, and multi-signature wallets.

When deploying contracts, you should use the latest released version of Solidity. This is because

# Les langages de développement de la blockchain

---

- ▶ **Solidity** – <https://solidity.readthedocs.io/en/latest/>
  - ▶ Langage "contrat intelligent" Ethereum de référence (au moins aujourd'hui)
- ▶ **Bamboo** – <https://github.com/CornellBlockchain/bamboo>
  - ▶ Un langage "contrat intelligent" en train de se transformer
- ▶ **Vyper** – <https://vyper.readthedocs.io/en/latest/>
  - ▶ Nouveau langage de programmation pythonique (Python) expérimental
- ▶ **Flint** – <https://github.com/flintlang/flint>
  - ▶ Nouveau langage en cours de développement avec des fonctionnalités de sécurité comprenant des types d'actif, une transition d'état et des entiers sécurisés

# Introduction à Solidity

---

- ▶ **Solidity est le langage le plus utilisé pour la rédaction de contrats intelligents avec Ethereum.** <https://solidity.readthedocs.io/en/v0.5.6/>
  - ▶ La programmation de base de Solidity est assez facile à apprendre.
  - ▶ Semblable à JavaScript, il possède cependant certaines fonctionnalités des langages orientés objet tels que Java et C ++.
- ▶ **Pour apprendre la programmation Solidity, avez juste besoin des éléments suivants:**
  - ▶ Un navigateur web (Chrome, Firefox, Edge, Opera, etc.)
  - ▶ Un connexion Internet
  - ▶ L'environnement de développement de Solidity est « Remix » et il est disponible en ligne <http://remix.ethereum.org>

# Environnement de développement

---

- ▶ On peut aussi utiliser des outils comme VS Code, Atom, Sublime Text, etc.
- ▶ Pour développer des applications décentralisées avec Ethereum, il faut au minimum avoir les outils suivants:
  - ▶ Nodejs <https://nodejs.org/en/>
  - ▶ Un compilateur Solidity, sous la forme d'un package npm : **npm install -g solc**
  - ▶ Un éditeur de texte, comme Visual Studio Code <https://code.visualstudio.com/>
  - ▶ Un client Ethereum pour développeurs, Ganach-cli : **npm install -g ganache-cli**
  - ▶ <https://truffleframework.com/ganache>
    - ▶ L'ancienne version obsolète s'appelle « ethereumjs-testrpc »
- ▶ D'autres outils avancés sont disponible dans le framework truffle
  - ▶ <https://truffleframework.com>

# Fichier dans Solidity

---

- ▶ **Extension :**
  - ▶ Un fichier Solidity est toujours enregistré avec une extension .sol.
- ▶ **Stockage du fichier**
  - ▶ Dans Remix, le fichier est toujours stocké dans la cache du navigateur
  - ▶ On peut aussi l'enregistrer sur disque et le charger
- ▶ **Le fichier (ou les fichiers) Solidity représente le « Smart Contract » de votre application Dapp dans la blockchain**
- ▶ **Chaque « smart contrat » une fois compilé est formé de deux éléments:**
  - ▶ L'interface ABI :Application Binary Interface
  - ▶ le ByteCode (en analogie avec les fichiers .class de Java)

# Application Binary Interface

---

- ▶ Chaque smart contrat comporte une interface ABI
- ▶ Elle ressemble beaucoup à une API qui fonctionne entre un langage de haut niveau (Java, C, C++) et le code binaire (Assembleur) de la couche matérielle.
- ▶ L'ABI comprend les éléments suivants:
  - ▶ Tous les noms de fonctions
  - ▶ Tous types de fonctions d'entrée et de sortie
  - ▶ Tous les noms d'événements et leurs paramètres

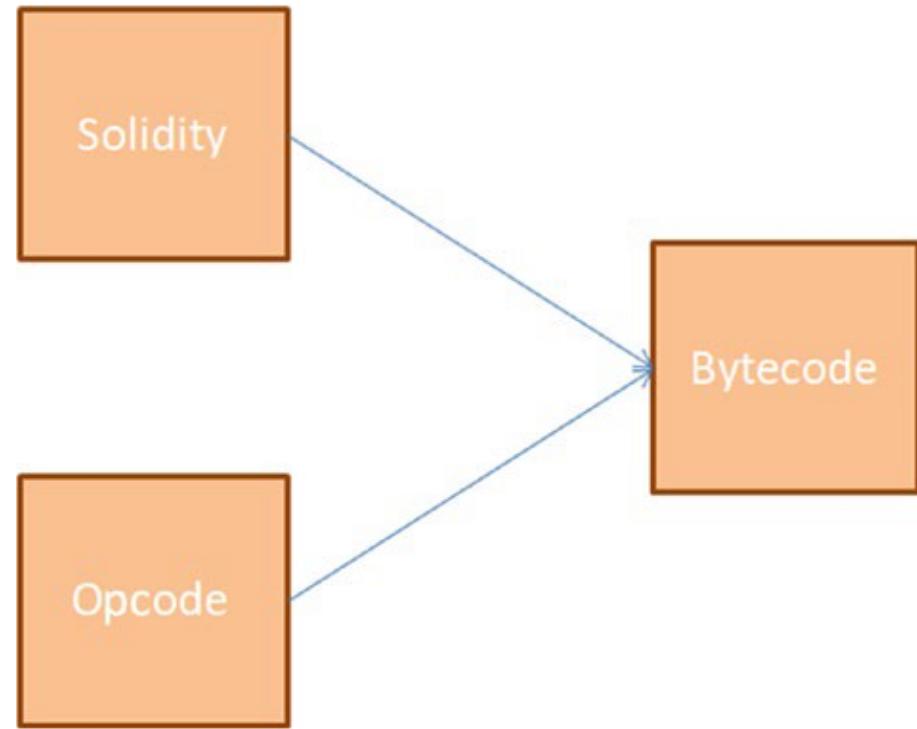
# Exemple de ABI

---

```
{  
    "constant": false,  
    "inputs": [],  
    "name": "beforeAll",  
    "outputs": [],  
    "payable": false,  
    "stateMutability": "nonpayable",  
    "type": "function"  
},  
  
{  
    "constant": false,  
    "inputs": [],  
    "name": "checkWinningProposal",  
    "outputs": [],  
    "payable": false,  
    "stateMutability": "nonpayable",  
    "type": "function"  
}
```

# ByteCode de Solidity

- ▶ Solidity est comme Java, il utilise une machine virtuelle Ethereum (EVM)
- ▶ EVM exécute des programmes compilés sous forme de « bytecode » Solidity



# Exemple de ABI + ByteCode

```
var test3Contract = web3.eth.contract([{"constant":true,"inputs":[],"name":"checkWinninPropc
var test3 = test3Contract.new(
{
    from: web3.eth.accounts[0],
    data: '0x608060405234801561001057600080fd5b50610878806100206000396000f3fe60806040526004
    gas: '4700000'
}, function (e, contract){
    console.log(e, contract);
    if (typeof contract.address !== 'undefined') {
        console.log('Contract mined! address: ' + contract.address + ' transactionHash: ' +
    }
})

```

# Syntaxe de Solidity

## ▶ Déclaration d'importation

- ▶ Vous pouvez importer d'autres fichiers au sein d'un fichier.
- ▶ Vous devez également spécifier le chemin du fichier correctement

```
import "remix_tests.sol"; // this import is automatically injected by Remix.  
import "./ballot.sol";
```

## ▶ Version

- ▶ La première ligne de code d'un fichier Solidity commence toujours par une annotation « `pragma` » dans laquelle vous corrigez le compilateur avec une version particulière.
- ▶ Si la version de Solidity est mise à jour, cela entraînera des problèmes d'incompatibilité

```
pragma solidity ^0.4.0;  
contract Ballot{  
}
```

```
1  pragma solidity >=0.4.22 <0.6.0;  
2  contract Ballot {  
3  }
```

# Exemple Solidity trivial

```
pragma solidity 0.4.19;

//contract definition
contract generalStructure {
    //state variables
    int public stateIntVariable; // variable of integer type
    string stateStringVariable; //variable of string type
    address personIdentifier; // variable of address type
    myStruct human; // variable of structure type
    bool constant hasIncome = true; //variable of constant nature

    //structure definition
    struct myStruct {
        string name; //variable fo type string
        uint myAge; // variable of unsigned integer type
        bool isMarried; // variable of boolean type
        uint[] bankAccountsNumbers; // variable - dynamic array of unsigned integer
    }

    //modifier declaration
    modifier onlyBy(){
        if (msg.sender == personIdentifier) {
            _;
        }
    }

    // event declaration
    event ageRead(address, int );

    //enumeration declaration
    enum gender {male, female}

    //function definition
    function getAge (address _personIdentifier) onlyBy() payable external returns (uint) {
        human = myStruct("Ritesh",10,true,new uint[](3)); //using struct myStruct
        gender _gender = gender.male; //using enum
        ageRead(personIdentifier, stateIntVariable);
    }
}
```

- ▶ Solidity fournit les multiples types de données prêts à l'emploi suivants:
  - ▶ bool
  - ▶ uint/int
  - ▶ bytes
  - ▶ address
  - ▶ Mapping
  - ▶ enum
  - ▶ struct
  - ▶ bytes/String

# Syntaxe de Solidity

```
//structure definition
struct myStruct {
    string name; //variable fo type string
    uint myAge; // variable of unsigned integer type
    bool isMarried; // variable of boolean type
    uint[] bankAccountsNumbers; // variable - dynamic array of unsigned integer
}
```

```
// event declaration
event ageRead(address, int );
```

```
human = myStruct("Ritesh",10,true,new uint[](3)); //using struct myStruct
```

```
//modifier declaration
modifier onlyBy(){
    if (msg.sender == personIdentifier) {
        -
    }
}
```

```
//function definition
function getAge (address _personIdentifier) onlyBy() payable external returns (uint) {
    human = myStruct("Ritesh",10,true,new uint[](3)); //using struct myStruct
    gender _gender = gender.male; //using enum
    ageRead(personIdentifier, stateIntVariable);
}
```

# Les fonctions de vue, constante et pure

```
pragma solidity ^0.4.17;           pragma solidity ^0.4.17;

contract ViewFunction {           contract PureFunction {

    function GetTenerValue(int _data) public view returns (int) {
        return _data * 10;
    }
}

function SimpleTransferToAccount() public {
    msg.sender.transfer(1);
}

function SimpleSendToAccount() public returns (bool) {
    return msg.sender.send(1);
}
```

# Les exceptions dans Solidity

```
pragma solidity ^0.4.19;

contract AssertContract {

    function ValidInt8(uint _data) public returns(uint8){
        require(_data >= 0);
        require(_data <= 255);

        uint8 value = 20;

        //checking datatype overflow
        assert (value + _data <= 255);

        return uint8(value + _data);
    }

    event LogFunctionFlow(string);

    function ValidInt8(int _data) public returns(uint8){
        LogFunctionFlow("Within function ValidInt8");

        if(_data < 0 || _data > 255) {
            revert();
        }

        LogFunctionFlow("Value is within expected range");
        LogFunctionFlow("Returning value from function");

        return uint8(_data);
    }
}
```

# Gestion des exceptions

```
logs [ {  
    "topic": "b5b850034705238ab6360bdc803e9e3dcaaf926c812b20193e2e99a5918d47b0",  
    "event": "LogFunctionFlow",  
    "args": [  
        "Within function ValidInt8"  
    ]  
},  
{  
    "topic": "b5b850034705238ab6360bdc803e9e3dcaaf926c812b20193e2e99a5918d47b0",  
    "event": "LogFunctionFlow",  
    "args": [  
        "Value is within expected range"  
    ]  
},  
{  
    "topic": "b5b850034705238ab6360bdc803e9e3dcaaf926c812b20193e2e99a5918d47b0",  
    "event": "LogFunctionFlow",  
    "args": [  
        "Returning value from function"  
    ]  
}  
]
```

## Regarder des événements individuels

```
var myEvent = instance.allEvents({fromBlock: 24000, toBlock: 'latest'});
myEvent.watch(function(error, result){
  if(error) {
    console.log(error);
  }
  console.log(result)
});
```

```
{ address: '0x600c320dd768fb55f03748d4d4028db2caf06a9',
  blockNumber: 24864,
  transactionHash: '0x38ca3d4b40f8e75d27ab3950234d837e96dbdd086178f139c4e675dc6531ee15',
  transactionIndex: 0,
  blockHash: '0x778b9a9a89b4609475dcc52d4c60de44254c24f8356b4886496488f57761ca0c',
  logIndex: 0,
  removed: false,
  event: 'ageRead',
  args: { '' : '33' } }
```

# Déploiement de Smart Contrat

- ▶ **Etape 1:**
    - ▶ Allez à <http://remix.ethereum.org> pour déployer votre smart contract dans la blockchain
    - ▶ Vous allez ensuite créer une transaction spéciale pour déployer le contrat dans une adresse spéciale, dite « l'adresse zero » 0x000
    - ▶ L'adresse zéro est une adresse spéciale qui indique à la blockchain Ethereum que vous souhaitez enregistrer un contrat.
  - ▶ **Etape 2: Remix se chargera de tout cela et envoie la transaction à MetaMask.**
    - ▶ Run/Injected Web3 (il faut avoir Remix et MetaMask sur ropsten)
    - ▶ **Afficher l'inspection (inspecter) dans chrome et tapez dans la console :  
`window.ethereum.enable();` (Sinon ça ne marchera pas)**
    - ▶ Vérifier l'adresse du smart contrat et déployer ensuite le smart contrat dans la blockchain Ropsten (ça pourrait ne pas fonctionner avec JSON API 2.0 !!)
    - ▶ Tester avec <http://remix-alpha.ethereum.org>

# Déploiement de Smart Contrat

Faucet at 0x72e...c7829 (blockchain)

(fallback)

withdraw uint256 withdraw\_amount



CONFIRM TRANSACTION Ropsten Test Net

Account 1  
9E7139..124F  
1.001 ETH  
996.78 USD

New Contract

Amount 0 ETH  
0.00 USD

Gas Limit 113558 UNITS

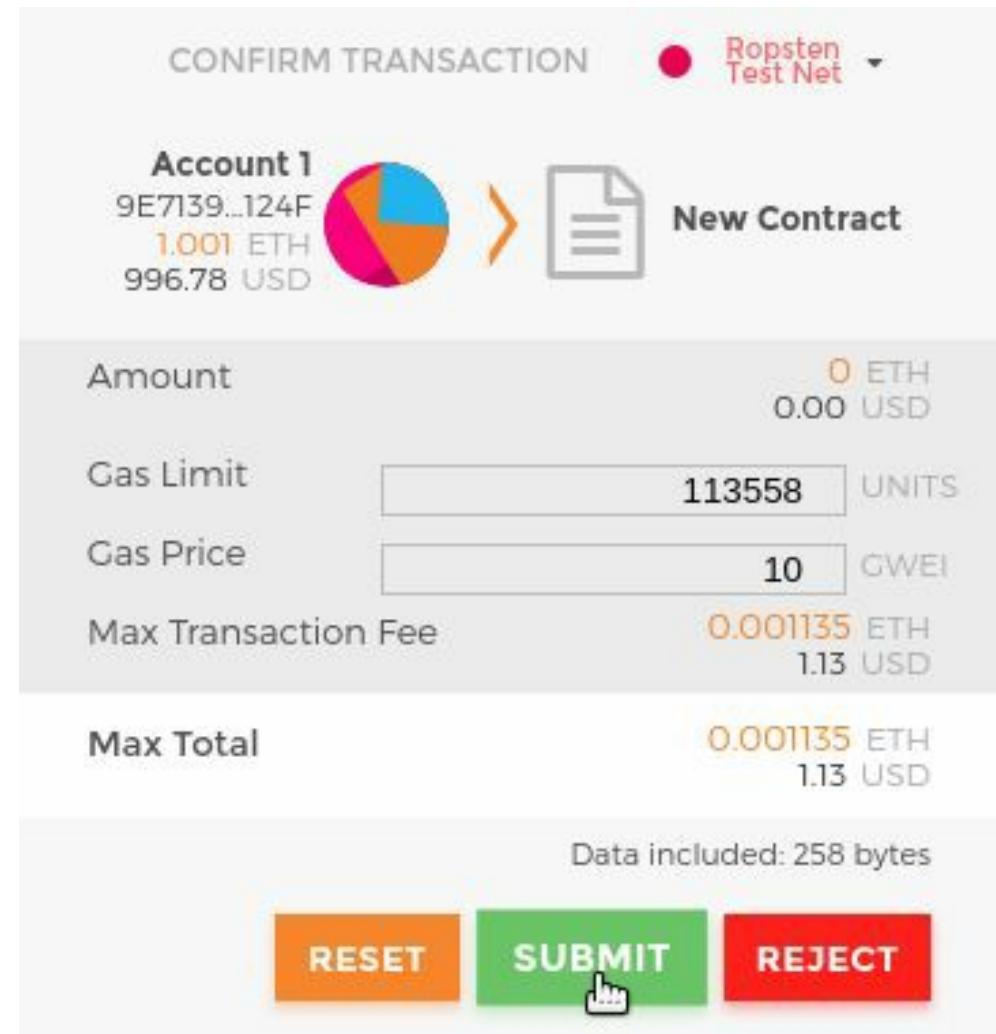
Gas Price 10 GWEI

Max Transaction Fee 0.001135 ETH  
1.13 USD

Max Total 0.001135 ETH  
1.13 USD

Data included: 258 bytes

RESET SUBMIT REJECT





# Déploiement de Smart Contrat

Etherscan ROPSTEN  
The Ethereum Block Explorer

ROPSTEN (Revival) TESTNET  Search by Address / Txhash / BlockNo

HOME BLOCKCHAIN ACCOUNT TOKEN CHART MISC

Contract Address 0x72E9D27f206fD62eaC5B81129aa3e774015c7829 [Home](#) / [Contract Accounts](#) / [Address](#)

---

Contract Overview		Misc	
ETH Balance:	0 Ether	Contract Creator:	0x9e713963a92c... at txn <a href="#">0x90333f7ecc9d...</a>
No Of Transactions:	1 txn		

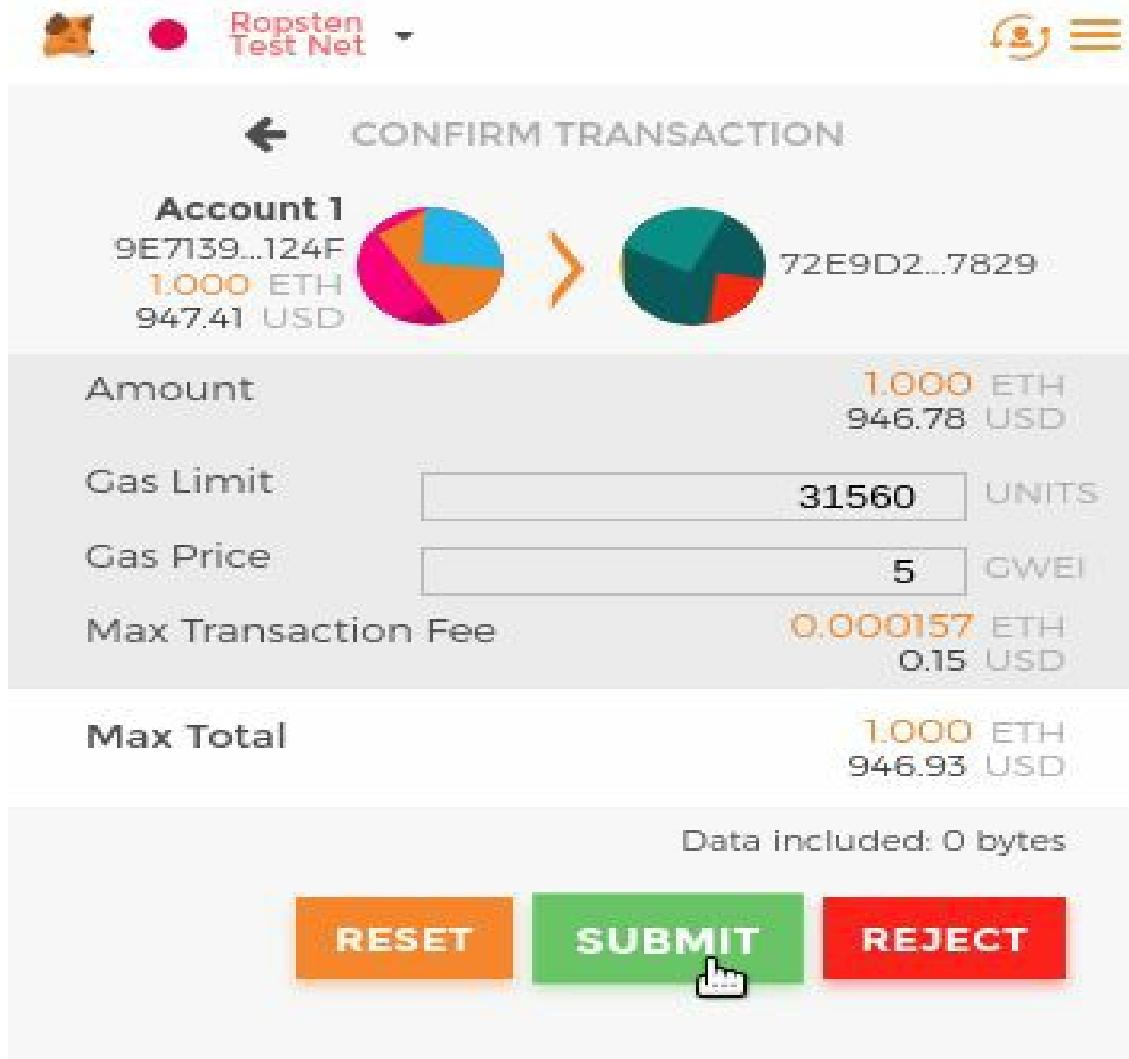
[Transactions](#) [Contract Code](#)

Latest 1 txn

TxHash	Block	Age	From	To	Value	[TxFee]
0x90333f7ecc9d...	2567995	16 hrs 48 mins ago	0x9e713963a92c...	IN Contract Creation	0 Ether	0.00113558

[ Download CSV Export ]

# Alimenter le Smart Contract



▶ Lorsque le smart contrat utilise une fonction publique « payable », il peut recevoir de la crypto-monnaie à partir d'un autre compte ou d'une autre Dapp de manière autonome

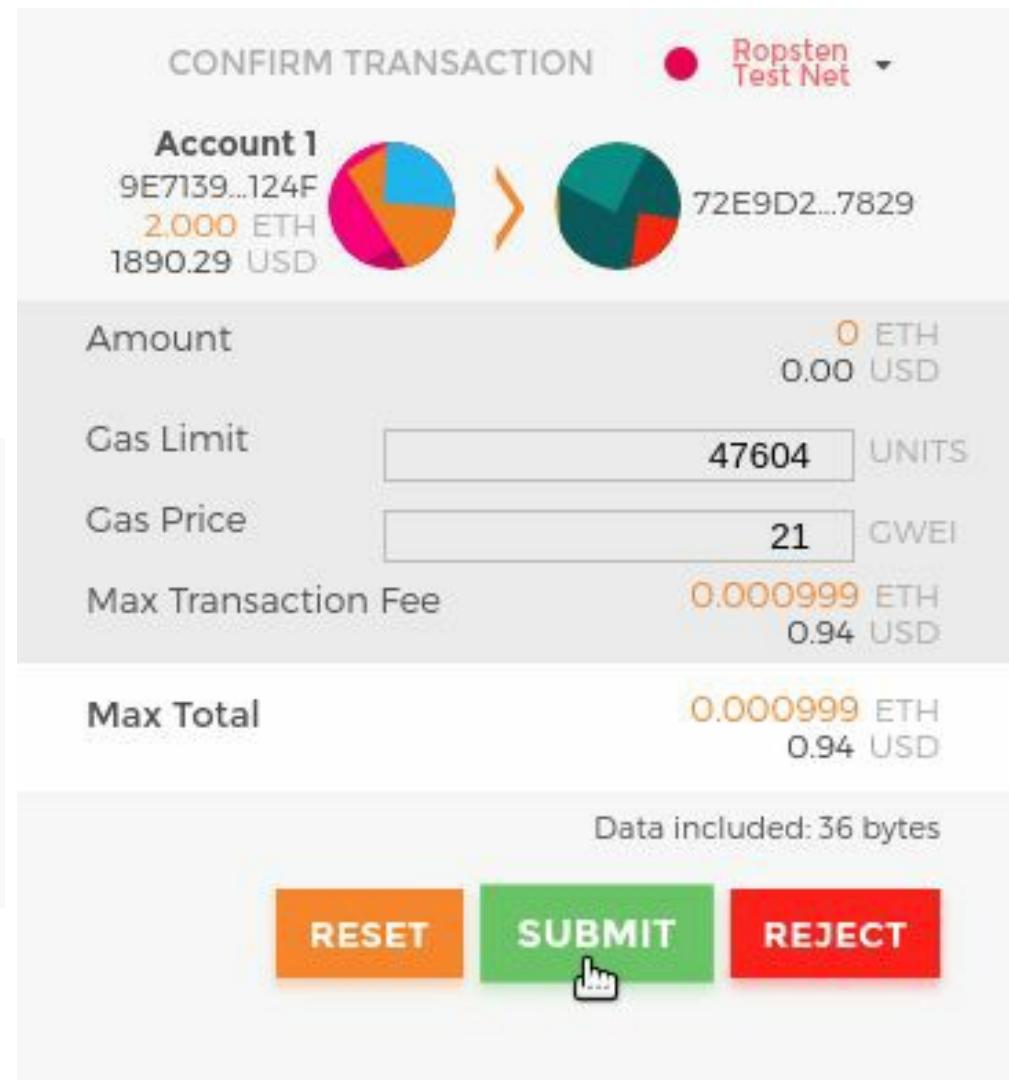
```
// It is important to also provide the
// `payable` keyword here, otherwise the function will
// automatically reject all Ether sent to it.
function register() public payable costs(price) {
    registeredAddresses[msg.sender] = true;
}

function changePrice(uint _price) public onlyOwner {
    price = _price;
}
```

# Fonction retrait du smart contrat

- ▶ Transaction MetaMask pour appeler la fonction de retrait

```
/// @notice Withdraw ether from bank
/// @return The balance remaining for the user
function withdraw(uint withdrawAmount) public returns (uint remainingBal) {
    // Check enough balance available, otherwise just return balance
    if (withdrawAmount <= balances[msg.sender]) {
        balances[msg.sender] -= withdrawAmount;
        msg.sender.transfer(withdrawAmount);
    }
    return balances[msg.sender];
}
```



# Etherscan affiche la transaction en appelant la fonction de retrait

 **Etherscan**  
ROPSTEN  
The Ethereum Block Explorer

ROPSTEN (Revival) TESTNET  **GO**

HOME BLOCKCHAIN **ACCOUNT** TOKEN CHART MISC

Contract Address [0x72E9D27f206fD62eaC5B81129aa3e774015c7829](#) [Home](#) / [Contract Accounts](#) / [Address](#)

---

Contract Overview		Misc	More Options	
ETH Balance:	0.9 Ether	Contract Creator	<a href="#">0x9e713963a92c...</a> at txn <a href="#">0x90333f7ecc9d...</a>	
No Of Transactions:	3 txns			

**Transactions** Internal Transactions Contract Code

Latest 3 txns

TxHash	Block	Age	From	To	Value	[TxFee]
0x3641e33d64dc...	2574307	2 mins ago	0x9e713963a92c...	IN	0x72e9d27f20...	0 Ether 0.000619038
0xebdc3c2ac500...	2574232	18 mins ago	0x9e713963a92c...	IN	0x72e9d27f20...	1 Ether 0.0003156
0x90333f7ecc9d...	2567995	17 hrs 58 mins ago	0x9e713963a92c...	IN	Contract Creation	0 Ether 0.00113558

[ Download CSV Export  ]

# Les clients Ethereum

---

- ▶ **Les plus utilisés**
  - ▶ Parité, écrit en Rust
  - ▶ Geth, écrit en Go
- ▶ **Les autres client (peuvent interagir et s'intégrer dans votre code)**
  - ▶ cpp-ethereum, écrit en C ++
  - ▶ pyethereum, écrit en python
  - ▶ Mantis, écrit en scala
  - ▶ Harmony, écrit en Java

# Le client Geth (<https://geth.ethereum.org/downloads/>)

Go Ethereum    Install    Downloads    Documentation

## Download Geth – Xircus (v1.8.23) – [Release Notes](#)

You can download the latest 64-bit stable release of Geth for our primary platforms below. Packages for all supported platforms, as well as develop builds, can be found further down the page. If you're looking to install Geth and/or associated tools via your favorite package manager, please check our [installation guide](#).

 [Geth 1.8.23 for Linux](#)

 [Geth 1.8.23 for macOS](#)

 [Geth 1.8.23 for Windows](#)

 [Geth 1.8.23 sources](#)

## Specific Versions

If you're looking for a specific release, operating system or architecture, below you will find:

- All stable and develop builds of Geth and tools
- Archives for non-primary processor architectures
- Android library archives and iOS XCode frameworks

Please select your desired platform from the lists below and download your bundle of choice. Please be aware that the [MD5](#) checksums are provided by our binary hosting platform (Azure Blobstore) to help check for download errors. **For security guarantees please verify any downloads via the attached PGP signature files** (see [OpenPGP](#))

# Execution de Geth (Synchronisation avec Blockchain)

```
invite de commandes - geth
Microsoft Windows [version 10.0.17763.379]
(c) 2018 Microsoft Corporation. Tous droits réservés.

C:\Users\hakir>geth
INFO [04-03|12:56:44.994] Maximum peer count
INFO [04-03|12:56:45.368] Starting peer-to-peer node
INFO [04-03|12:56:45.376] Allocated cache and file handles
INFO [04-03|12:56:45.598] Writing default main-net genesis block
INFO [04-03|12:56:45.983] Persisted trie from memory database
INFO [04-03|12:56:45.994] Initialised chain configuration
EIP158: 2675000 Byzantium: 4370000 Constantinople: 7280000 ConstantinopleFix: 7280000 Engine: ethash
INFO [04-03|12:56:46.009] Disk storage enabled for ethash caches
INFO [04-03|12:56:46.017] Disk storage enabled for ethash DAGs
INFO [04-03|12:56:46.026] Initialising Ethereum protocol
INFO [04-03|12:56:46.049] Loaded most recent local header
INFO [04-03|12:56:46.057] Loaded most recent local full block
INFO [04-03|12:56:46.065] Loaded most recent local fast block
INFO [04-03|12:56:46.089] Regenerated local transaction journal
INFO [04-03|12:56:46.255] New local node record
INFO [04-03|12:56:46.264] Started P2P networking
2bd856dd4b30967c5901c158819c73e1fa392d0b@127.0.0.1:30303
INFO [04-03|12:56:46.265] IPC endpoint opened
INFO [04-03|12:57:24.412] New local node record
ETH=25 LES=0 total=25
instance=Geth/v1.8.23-stable-c9427004/windows-amd64/go1.11.5
database=C:\Users\hakir\AppData\Roaming\Ethereum\geth\chaindata cache=512 handles=8192
nodes=12356 size=1.88mB time=72.0019ms gcnodes=0 gctime=0s livenodes=1 livesize=0.00B
config="{ChainID: 1 Homestead: 1150000 DAO: 1920000 DAOSupport: true EIP150: 2463000 EIP155: 2675000
dir=C:\Users\hakir\AppData\Roaming\Ethereum\geth\ethash count=3
dir=C:\Users\hakir\AppData\Ethash count=2
versions="[63 62]" network=1
number=0 hash=d4e567...cb8fa3 td=17179869184 age=49y11mo2w
number=0 hash=d4e567...cb8fa3 td=17179869184 age=49y11mo2w
number=0 hash=d4e567...cb8fa3 td=17179869184 age=49y11mo2w
transactions=0 accounts=0
seq=1 id=06e81c222a4fcf2b ip=127.0.0.1 udp=30303 tcp=30303
self=enode://68b197ac9f27b8de05264ffe1dae83482d46bba700127021c0ae195ce0a9dc8dda874c1333dc2fdff945b40a
url=\.\.\.\pipe\geth.ipc
seq=2 id=06e81c222a4fcf2b ip=196.229.192.66 udp=30303 tcp=30303
```

# Geth help

```
cmd Invite de commandes
NAME:
  geth - the go-ethereum command line interface

  Copyright 2013-2018 The go-ethereum Authors

USAGE:
  geth [options] command [command options] [arguments...]

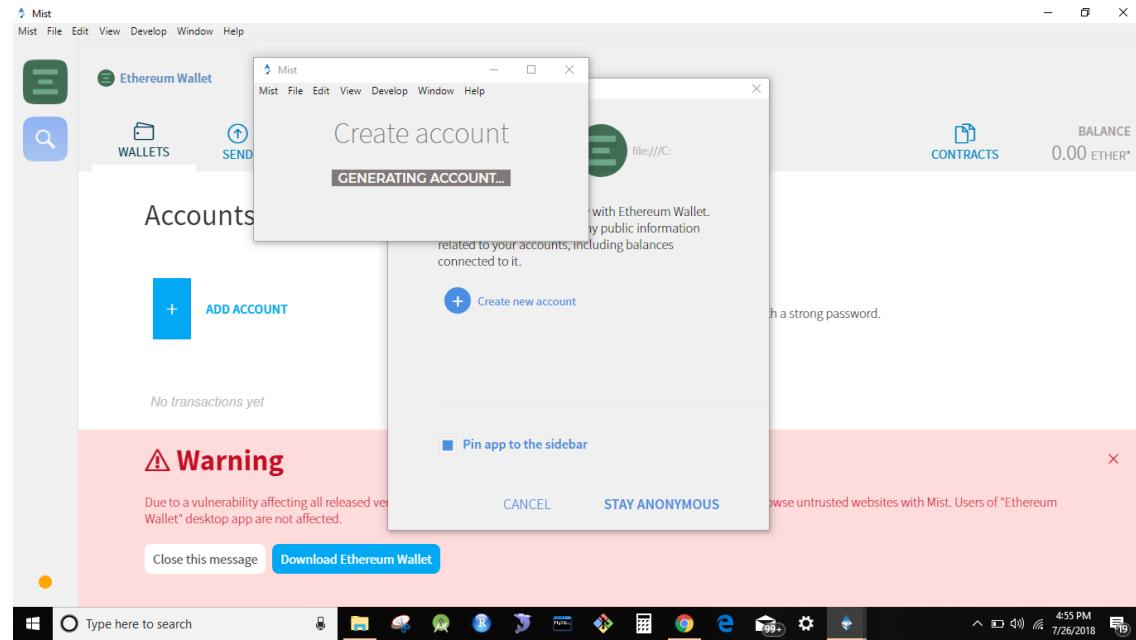
VERSION:
  1.8.23-stable-c9427004

COMMANDS:
  account          Manage accounts
  attach           Start an interactive JavaScript environment (connect to node)
  bug              opens a window to report a bug on the geth repo
  console          Start an interactive JavaScript environment
  copydb           Create a local chain from a target chaindata folder
  dump             Dump a specific block from storage
  dumpconfig       Show configuration values
  export            Export blockchain into file
  export-preimages Export the preimage database into an RLP stream
  import            Import a blockchain file
  import-preimages Import the preimage database from an RLP stream
  init              Bootstrap and initialize a new genesis block
  js                Execute the specified JavaScript files
  license           Display license information
  makecache         Generate ethash verification cache (for testing)
  makedag           Generate ethash mining DAG (for testing)
  monitor           Monitor and visualize node metrics
  removedb         Remove blockchain and state databases
  version           Print version numbers
  wallet            Manage Ethereum presale wallets
  help, h           Shows a list of commands or help for one command

ETHEREUM OPTIONS:
  --config value      TOML configuration file
  --datadir "C:\Users\hakir\AppData\Roaming\Ethereum" Data directory for the databases and keystore
  --keystore          Directory for the keystore (default = inside the datadir)
  --nousb             Disables monitoring for and managing USB hardware wallets
  --networkid value   Network identifier (integer, 1=Frontier, 2=Morden (disused), 3=Ropsten, 4=Rinkeby) (default: 1)
  --testnet            Ropsten network: pre-configured proof-of-work test network
  --rinkeby           Rinkeby network: pre-configured proof-of-authority test network
  --goerli            GÖERLI network: pre-configured proof-of-authority test network
  -- Suite --
```

# Geth GUI

## Mist Browser : obsolète



## MyEtherWallet | MEW

The screenshot shows the MyEtherWallet (MEW) website. The header includes the MEW logo, navigation links for 'Home', 'About', 'FAQs', and a language selector for 'English'. The main content features a large button labeled 'Get a New Wallet' with sub-options: 'MEWconnect' (selected), 'By Keystore File', and 'By Mnemonic Phrase'. Below this, a section titled 'Recommended Method !' describes a 'Safe and easy access to your wallet in 3 steps.' Buttons for 'Download on the App Store' and 'GET IT ON Google Play' are shown, along with a 'Scan to Download' link.

# Le client Parity (<https://www.parity.io/ethereum/>)

The screenshot shows the official website for Parity Ethereum. At the top left is the Parity logo, which consists of a stylized diamond shape made of horizontal lines next to the word "parity". Along the top navigation bar are links for TECHNOLOGIES, SOLUTIONS, ABOUT, and BLOG, along with social media icons for Twitter and GitHub. The main title "parity ethereum" is centered above a subtitle "The fastest and most advanced Ethereum client." A prominent red button labeled "DOWNLOAD PARITY ETHEREUM" is positioned below the subtitle. Below this button, a link says "or [learn more](#) about Parity Ethereum". At the bottom of the page are three call-to-action boxes: one for GitHub (latest changes and version history), one for documentation (complete Parity Ethereum documentation), and one for community (connect with support and the Parity Ethereum community).

parity

TECHNOLOGIES SOLUTIONS ABOUT BLOG

parity ethereum

The fastest and most advanced Ethereum client.

DOWNLOAD PARITY ETHEREUM

or [learn more](#) about Parity Ethereum

See the latest changes and complete version history.

Read the complete Parity Ethereum documentation.

Connect with support and the Parity Ethereum community.

# Le client Parity

(<https://www.parity.io/ethereum/#download>)

## Download Parity Ethereum

**Stable**

Stable is the most mature and tested version of Parity Ethereum

 [Download Stable](#)

**Beta**

MOST POPULAR

Beta comes with additional features and better performance but may yet have quirks and issues to be fixed

 [Download Beta](#)

**Nightly**

Nightly is a cutting-edge software build but is not recommended for managing anything of value

 [Get the Code](#)

# Parity Ethereum 2.3.9-stable

System	Architecture	Binary	SHA256 Checksum
	x64	<a href="#">parity.exe</a>	<a href="#">716cd3f23efedca39d9a3fc959506e72627cdd2891ef7f37dea8268a923c7164</a>
	x64	<a href="#">parity</a>	<a href="#">03b6f4cc6d5b063fa18b1576a993ad862d03026a193a0c45ee3e7cc390ee7816</a>
	x64	<a href="#">parity</a>	<a href="#">d4e3497caeee149ca43c49950c948b298d51f381eaa8276745df7c93ad7e677fc</a>
System	Option	-	Resource
	Homebrew	-	<a href="#">github.com/paritytech/homebrew-paritytech</a>
	Snapcraft	-	<a href="#">snapcraft.io/parity</a>
	Docker	-	<a href="#">hub.docker.com/r/parity/parity</a>
	All binaries	-	<a href="#">vanity-service.parity.io/parity-binaries?version=v2.3.9</a>

# Execution de Parity (Synchronisation avec Blockchain)

```
C:\Users\hakir\Downloads\parity.exe
2019-04-03 12:53:32 Starting Parity-Ethereum/v2.3.9-stable-0b428262e-20190401/x86_64-windows-msvc/rustc1.33.0
2019-04-03 12:53:32 Keys path C:\Users\hakir\AppData\Roaming\Parity\Ethereum\keys\ethereum
2019-04-03 12:53:32 DB path C:\Users\hakir\AppData\Local\Parity\Ethereum\chains\ethereum\db\906a34e69aec8c0d
2019-04-03 12:53:32 State DB configuration: fast
2019-04-03 12:53:32 Operating mode: active
2019-04-03 12:53:34 Configured for Ethereum using Ethash engine
2019-04-03 12:53:34 Multi-threaded server is not available on Windows. Falling back to single thread.
2019-04-03 12:53:35 Updated conversion rate to E1 = US$168.12 (28324440 wei/gas)
2019-04-03 12:53:39 Public node URL: enode://a3fd46b1869bdfb36c6f6a9b14406c7df0d1c5f70a6f1fff76c7452af927ba1c339e60055ff036fd0a962e0f2b8f008c49623b0a6dc6448ebf0967a4c0
1193c3@127.0.0.1:30303
2019-04-03 12:53:39 Snapshot initializing (0 chunks restored)      #0    4/25 peers      8 KiB chain      3 MiB db   0 bytes queue      10 KiB sync   RPC: 0 conn, 0 req
/s, 0 µs
2019-04-03 12:53:44 Syncing snapshot 0/2917      #0    4/25 peers      8 KiB chain      3 MiB db   0 bytes queue      10 KiB sync   RPC: 0 conn, 0 req/s, 0 µs
2019-04-03 12:53:49 Syncing snapshot 0/2917      #0    4/25 peers      8 KiB chain      3 MiB db   0 bytes queue      10 KiB sync   RPC: 0 conn, 0 req/s, 0 µs
2019-04-03 12:53:54 Syncing snapshot 0/2917      #0    5/25 peers      8 KiB chain      3 MiB db   0 bytes queue      10 KiB sync   RPC: 0 conn, 0 req/s, 0 µs
2019-04-03 12:54:16 Syncing snapshot 2/2917      #0    7/25 peers      8 KiB chain      3 MiB db   0 bytes queue      10 KiB sync   RPC: 0 conn, 0 req/s, 0 µs
2019-04-03 12:54:16 Syncing snapshot 3/2917      #0    7/25 peers      8 KiB chain      3 MiB db   0 bytes queue      10 KiB sync   RPC: 0 conn, 0 req/s, 0 µs
2019-04-03 12:54:17 Syncing snapshot 4/2917      #0    7/25 peers      8 KiB chain      3 MiB db   0 bytes queue      10 KiB sync   RPC: 0 conn, 0 req/s, 0 µs
2019-04-03 12:54:17 Syncing snapshot 4/2917      #0    7/25 peers      8 KiB chain      3 MiB db   0 bytes queue      10 KiB sync   RPC: 0 conn, 0 req/s, 0 µs
2019-04-03 12:54:19 Syncing snapshot 7/2917      #0    7/25 peers      8 KiB chain      3 MiB db   0 bytes queue      10 KiB sync   RPC: 0 conn, 0 req/s, 0 µs
2019-04-03 12:54:24 Syncing snapshot 9/2917      #0    7/25 peers      8 KiB chain      3 MiB db   0 bytes queue      10 KiB sync   RPC: 0 conn, 0 req/s, 0 µs
2019-04-03 12:54:29 Syncing snapshot 12/2917      #0    7/25 peers      8 KiB chain      3 MiB db   0 bytes queue      10 KiB sync   RPC: 0 conn, 0 req/s, 0 µs
2019-04-03 12:54:34 Syncing snapshot 14/2917      #0    7/25 peers      8 KiB chain      3 MiB db   0 bytes queue      10 KiB sync   RPC: 0 conn, 0 req/s, 0 µs
2019-04-03 12:54:39 Syncing snapshot 17/2917      #0    7/25 peers      8 KiB chain      3 MiB db   0 bytes queue      10 KiB sync   RPC: 0 conn, 0 req/s, 0 µs
2019-04-03 12:55:00 Syncing snapshot 17/2917      #0    9/25 peers      8 KiB chain      3 MiB db   0 bytes queue      10 KiB sync   RPC: 0 conn, 0 req/s, 0 µs
2019-04-03 12:55:01 Syncing snapshot 18/2917      #0    9/25 peers      8 KiB chain      3 MiB db   0 bytes queue      10 KiB sync   RPC: 0 conn, 0 req/s, 0 µs
2019-04-03 12:55:01 Syncing snapshot 20/2917      #0    9/25 peers      8 KiB chain      3 MiB db   0 bytes queue      10 KiB sync   RPC: 0 conn, 0 req/s, 0 µs
2019-04-03 12:55:01 Syncing snapshot 21/2917      #0    9/25 peers      8 KiB chain      3 MiB db   0 bytes queue      10 KiB sync   RPC: 0 conn, 0 req/s, 0 µs
2019-04-03 12:55:04 Syncing snapshot 24/2917      #0    10/25 peers     8 KiB chain      3 MiB db   0 bytes queue      10 KiB sync   RPC: 0 conn, 0 req/s, 0 µs
```

## Atelier 1: Application Tirage au sort dans la blockchain

Objectif: Maitriser Remix et le déploiement de Smart Contrat en local

# Création du Smart contract avec Remix IDE

- ▶ Pour ouvrir un nouveau fichier dans l'éditeur de remix, cliquez sur l'icône Nouveau fichier en haut à gauche.
- ▶ Le nom du fichier est arbitraire, appelons-le « `Blocksplit.sol` »
- ▶ Après confirmation, un nouvel onglet de fichier vide s'ouvrira.
- ▶ La première ligne de chaque projet Solidity est la déclaration de pragma
  - ▶ `pragma solidity >=0.4.22 <0.6.0;`



# Création du Smart contract avec Remix IDE

---

- ▶ Le corps d'un contrat intelligent est placé entre des accolades ({et}) et le nom est précédé de l'expression contrat.
- ▶ 

```
contract Blocksplit {  
}  
}
```
- ▶ Variables & getters
- ▶ D'abord, nous allons enregistrer tous les participants au tirage au sort. Une liste est idéale pour quelque chose comme ça.
- ▶ `address[] players;` // Cela signifie "une liste d'adresses nommées players".
- ▶ « adresse » est un type de données dans Solidity et peut contenir une adresse littérale de la blockchain Ethereum
- ▶ Remarque: certains autres types de Solidity sont `uint`, `string`, `bool`, `address`, etc.

# Déclarer les joueurs

---

- ▶ Nous avons également besoin d'un moyen d'enregistrer les joueurs de manières uniques.
- ▶ Nous ne voulons pas qu'un joueur puisse participer au jeu plusieurs fois à partir de la même adresse. Déclarons une nouvelle variable pour cela.
  - ▶ mapping (address => bool) uniquePlayers;
  - ▶ Le mappage est un autre type de données dans Solidity.
  - ▶ Vous pouvez imaginer un mappage comme une table à deux colonnes, l'une étant les clés et l'autre les valeurs, comme ceci:

key	value
x	1
y	200
z	234121

# Déclarer les joueurs

- ▶ Dans cet exemple,
  - ▶ mapping (address => bool) uniquePlayers signifie «La clé de uniqueMapping est une adresse et sa valeur est un booléen».
  - ▶ La lecture d'un mappage se fait comme suit: uniquePlayers [adresse],
  - ▶ Nous pourrons donc sauvegarder les adresses dans ce mappage avec la valeur true, indiquant qu'elles ont déjà participé

key	value
0xE2B28F870336B4eAA0aCc73cE02757fcC428dC9	true
0x4da2e85d64bece663ccab06e89b970b6b077f22f	true
...	...

# Déclarer les joueurs gagnants

---

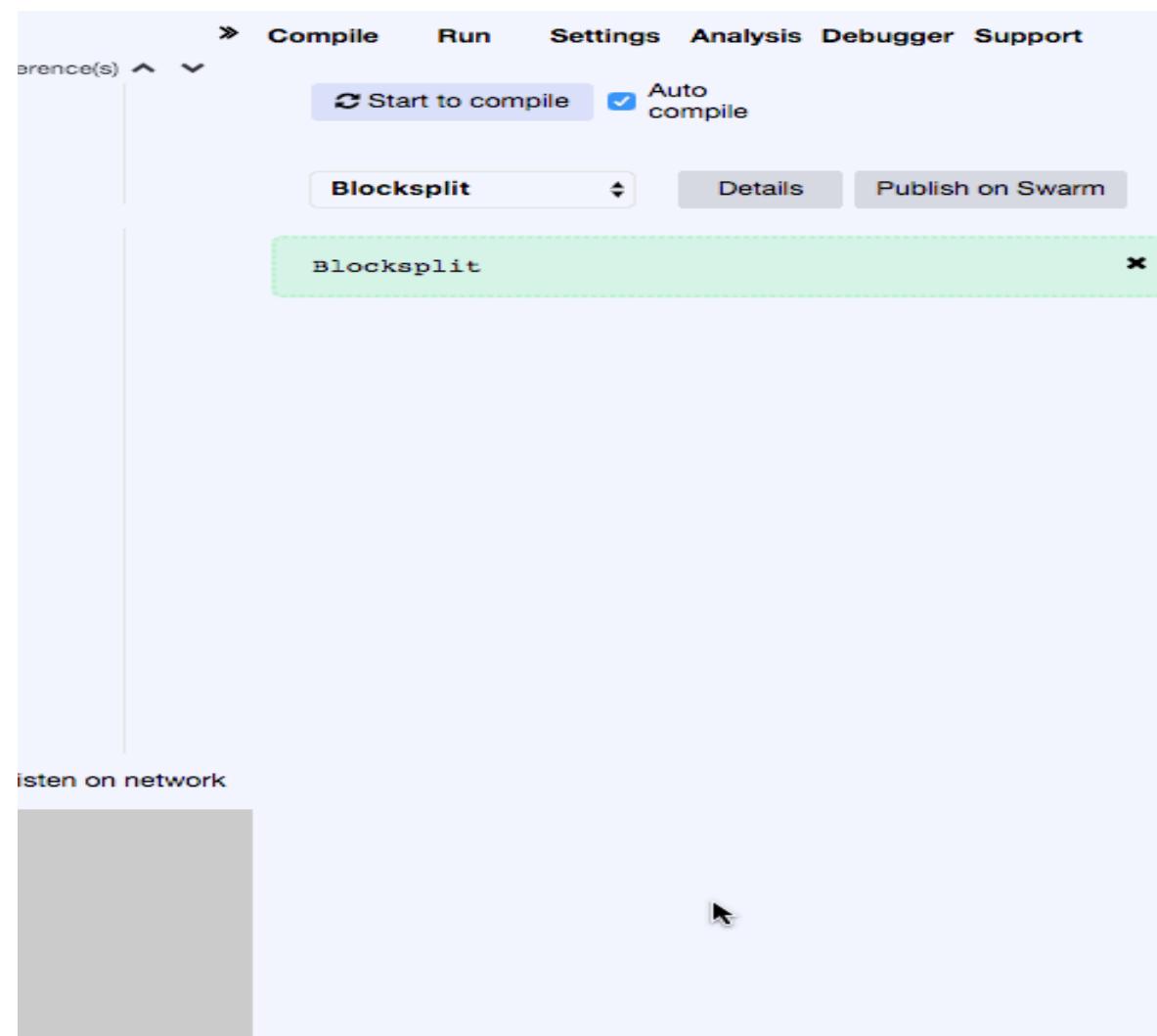
- ▶ Nous avons également besoin d'une variable qui permette aux gagnants d'économiser au moment du tirage au sort.
- ▶ Nous avons également besoin d'une variable dans laquelle nous enregistrerons l'adresse à laquelle les fonds seront envoyés à la fin du tirage au sort.
- ▶ Dans ce cas, ce sera l'adresse de Mining For Charity, car nous donnons tous les bénéfices à une association caritative: <http://mining4charity.eu/>

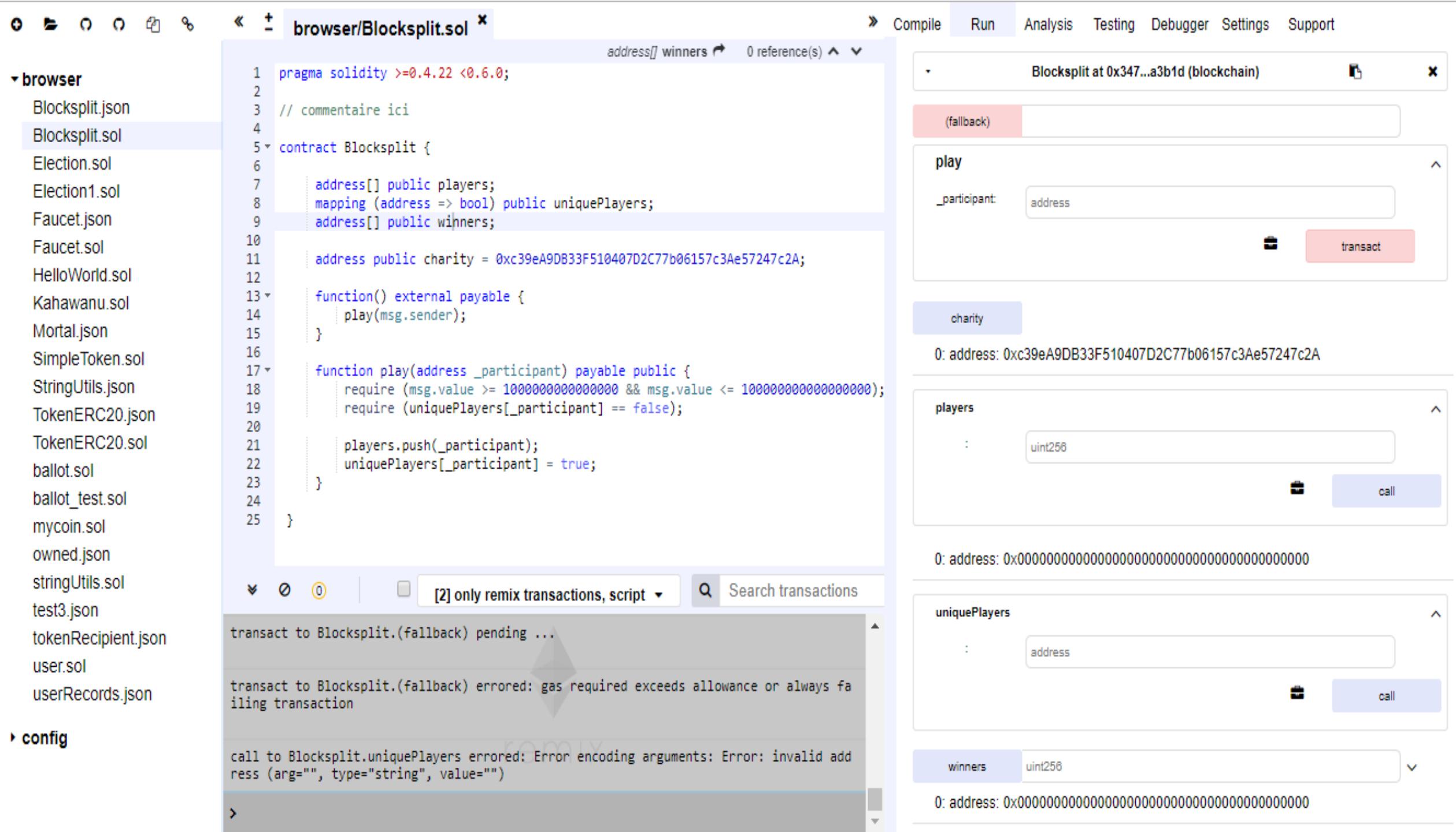
```
address[] public players;
mapping (address => bool) public uniquePlayers;
address[] public winners;

address public charity = 0xc39eA9DB33F510407D2C77b06157c3Ae57247c2A;
```

# Simuler le smart contrat

- ▶ Remix propose une blockchain locale « JavaScript VM » à votre navigateur pour vous permettre de simuler le comportement de votre smart contract
- ▶ Il est recommandé de tester votre application en local ou bien dans votre blockchain privée (TestRPC ou bien testbed) avant de déployer dans la blockchain Main !!!





# Fonction Fallback

---

- ▶ Pour qu'un contrat intelligent accepte les paiements, la fonction qui les accepte doit être déclarée comme étant « payable ».
  - ▶ `function() external payable {`
  - ▶     `play(msg.sender);`
  - ▶     `}`
- ▶ Une fonction Fallback est une fonction sans nom, sa déclaration commence par `function ()`.
- ▶ Le mot `external` signifie qu'il ne peut être appelé que de l'extérieur du contrat, et non de l'intérieur par d'autres fonctions.
- ▶ Le mot `payable` est important ici car il permet à la fonction de traiter les paiements à l'éther.
- ▶ Le corps de la fonction appelle une autre fonction appelée `play` et lui transmet un paramètre: « `msg.sender` ».

# Entrer dans la tombola

---

- ▶ Nos participants pourront s'inscrire au tirage au sort soit en envoyant de l'argent directement à l'adresse du tirage, soit en appelant la table de jeu et en fournissant l'adresse à laquelle ils souhaitent participer.
  - ▶ function play(address \_participant) payable public {
  - ▶     // Vérifier qu'on suffisamment de solde dans le compte (entre 0,001 Ether et 0,1)
  - ▶     require (msg.value >= 1000000000000000 && msg.value <= 1000000000000000);
  - ▶     require (uniquePlayers[\_participant] == false); // joueur n'est pas déjà dans la partie
  - ▶
  - ▶     players.push(\_participant);
  - ▶     uniquePlayers[\_participant] = true;
  - ▶ }
- ▶ La fonction est:
  - ▶ payable, ce qui signifie qu'elle peut accepter des transactions avec Ether attachée, et
  - ▶ publique, ce qui signifie qu'elle peut être appelée à la fois dans le contrat, comme le fait la fonction de fallback, et à l'extérieur, dans un autre contrat ou une interface portefeuille, comme MetaMask.

# Jouer dans la tombola

---

- ▶ Les deux fonctions suivantes sont utilisées pour chaque joueur
  - ▶ `players.push(_participant);`
  - ▶ `uniquePlayers[_participant] = true;`
- ▶ Rappelant qu'une liste a une fonction intégrée « `push` » pour ajouter un élément à la liste (ici, `_participant`)
- ▶ Les vérifications prétendantes « **require(...)** » sont vérifiées, un participant est ajouté à la liste
- ▶ Nous avons également défini sa valeur « **uniquePlayers** » sur « **true** » pour que la vérification échoue s'il tente à nouveau de participer
- ▶ Remarque: nous n'utilisons pas de boucle « `for` », parce qu'elle assez couteuses en « `gas price` »

# Editer le Smart Contract dans Remix

```
pragma solidity >=0.4.22 <0.6.0;

contract Blocksplit {
    address[] public players;
    mapping (address => bool) public uniquePlayers;
    address[] public winners;

    address public charity = 0xc39eA9DB33F510407D2C77b06157c3Ae57247c2A;

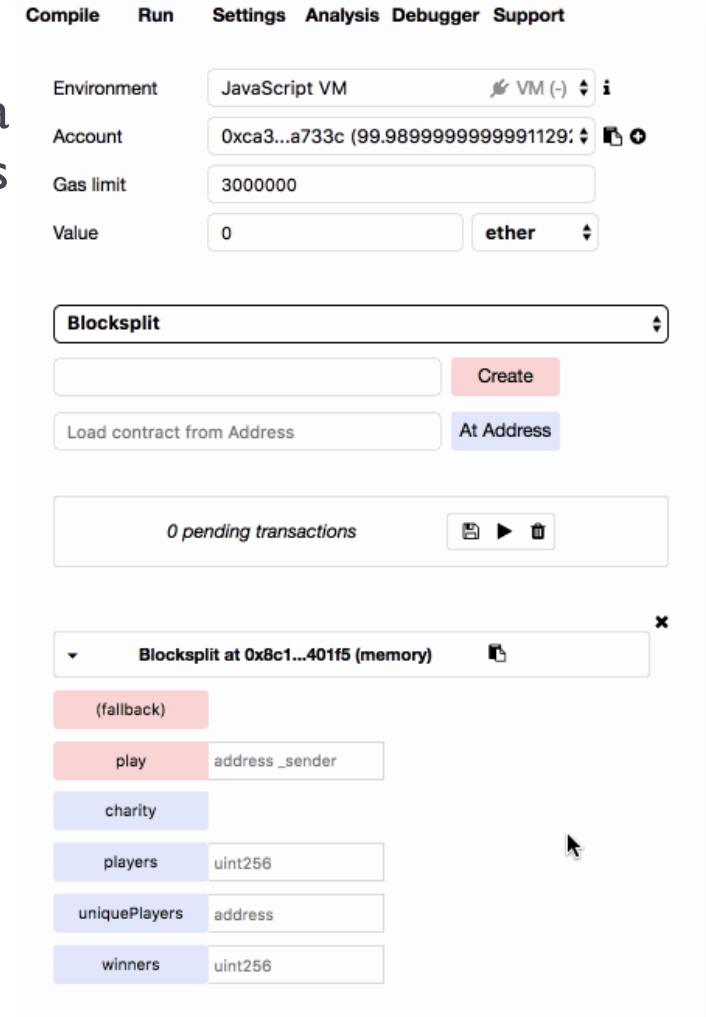
    function() external payable {
        play(msg.sender);
    }

    function play(address _participant) payable public {
        require (msg.value >= 1000000000000000 && msg.value <= 1000000000000000);
        require (uniquePlayers[_participant] == false);

        players.push(_participant);
        uniquePlayers[_participant] = true;
    }
}
```

# Examiner les fonctions dans Remix

- ▶ On constate deux fonctions en rouge: « fallback et play »
  - ▶ Celles-ci sont des fonctions d'écriture (elles changent l'état de la blockchain) et nécessitent une transaction, ce qui signifie qu'elles coûtent de l'argent pour être exécutées.
  - ▶ Utilisons l'option fallback pour envoyer de l'argent directement au contrat.
  - ▶ Premièrement, dans le menu « value », nous devons mettre un nombre compris entre 0,001 et 0,1.
  - ▶ Ensuite, l'unité de valeur (menu à droite de Valeur) doit être définie sur Ether.
  - ▶ Enfin, nous cliquons sur le bouton rouge de secours.
  - ▶ Si tout va bien, cliquer sur le bouton bleu des joueurs devrait afficher l'adresse à partir de laquelle nous venons d'exécuter la transaction.



# Examiner les fonctions dans Remix

- Ajoutez les 4 autres adresses que la machine virtuelle JavaScript générée pour vous dans le jeu en suivant la même procédure - il suffit de changer les adresses dans le menu du haut et d'exécuter le repli de chacune d'elles.
- Ensuite, parcourez la liste des joueurs en entrant les nombres ordinaux des joueurs ajoutés de 0 à 4 (0 en premier).
- Une fois le numéro 5 atteint, le programme commence à générer des erreurs car il n'y a que 5 joueurs dans le jeu (0, 1, 2, 3, 4, 5).



## Complément

---

- ▶ Tester les deux autres approches de déploiement de contrat Remix
  - ▶ **InjectedWeb3**, déploiement dans la blockchain **ropsten**.
  - ▶ **Web3 Provider**, déploiement du contrat dans une blockchain locale **Ganache**

# Remix : les alternatives

The screenshot shows the Truffle Suite website. At the top left is the Truffle logo (a stylized 'T' inside a circle). To its right is the text "TRUFFLE SUITE". Along the top navigation bar are links for "SUITE ▾", "DOCS", "TUTORIALS", "BOXES", "BLOG", "EVENTS", "COMMUNITY", and "UNIVERSITY". The main content area features a dark background with a central graphic. On the left, a semi-transparent box contains a snippet of Solidity smart contract code. In the center, there's a 3D-style illustration of three objects: a purple cube, a yellow cube, and a grey cylinder, all connected by glowing lines to a central point. To the right of the illustration is a large, curved text block: "SWEET TOOLS FOR SMART CONTRACTS". Below this, a smaller text block reads: "The Truffle Suite gets developers from idea to dapp as comfortably as possible." At the bottom left, the word "Truffle" is followed by a URL in green: <https://truffleframework.com/>. At the bottom right, a large green text block states: "C'est le couteau suisse Ethereum, il est le Framework le plus populaire d'Ethereum," with a small yellow downward arrow icon preceding the word "est".

Truffle: <https://truffleframework.com/>

C'est le couteau suisse Ethereum, il est le Framework le plus populaire d'Ethereum,

► 138 pour simplifier le déploiement de smart contrat

# Remix : les alternatives

The screenshot shows the homepage of the Embark website. The header features the Embark logo (a stylized rocket ship icon) and navigation links for Quick Start, Learn, Plugins, Community, and Blog. A search bar is positioned in the top right corner. The main visual is a dark space-themed background with a large orange planet and smaller celestial bodies. In the foreground, a small figure stands on a rocky outcrop. The central text reads "Embark into the Ether." followed by a subtitle: "The all-in-one developer platform for building and deploying decentralized applications." Below this are two buttons: "GET STARTED" and "npm install -g embark". A GitHub star icon with the number "2,734" is also visible. At the bottom, there is a green callout text: "Embark <https://embark.status.im/> un Framework qui vous permet de développer et de déployer facilement des applications décentralisées (DApps)."

Embark <https://embark.status.im/>  
un Framework qui vous permet de développer et de déployer  
facilement des applications décentralisées (DApps).

# Remix : les alternatives

## Ethereum language support in Atom

build passing

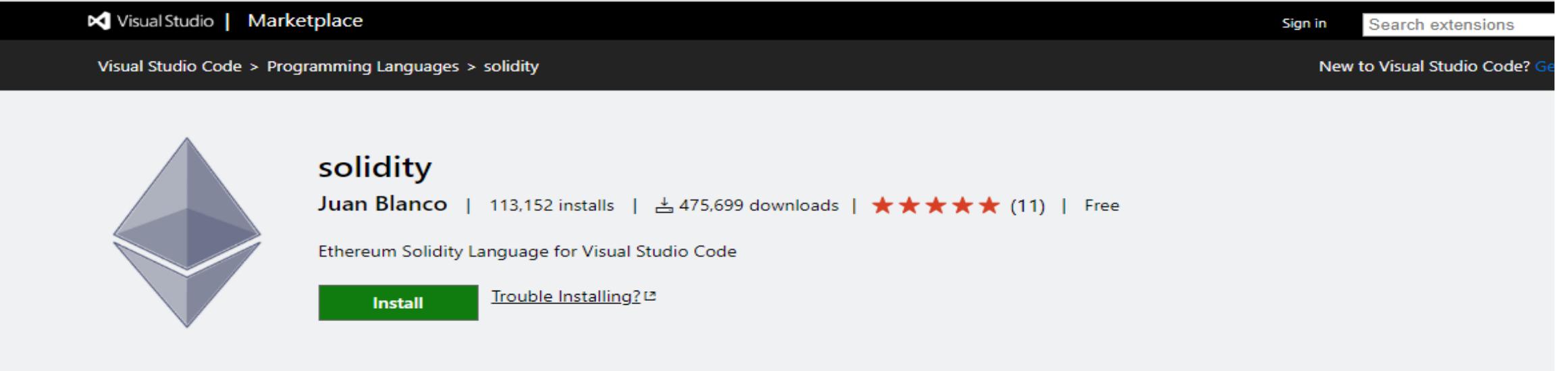
Adds syntax highlighting and snippets to Solidity and Serpent files in Atom.

Solidity support converted from the [Ethereum Sublime package](#).

```
8  #
9  # Example ETX subcurrency
10 #
11
12 data owner
13 data exchange
14 data market_id
15 data balances[2^160](balance)
16
17 extern exchange: [deposit:[int256,int256,int256]:int256]
18
19 def init():
20     self.owner = msg.sender
21     self.balances[msg.sender].balance = 1000000 * 10 ** 5
22
23 def transfer(recipient, amount):
24     # Prevent negative send from stealing funds
25     if recipient <= 0 or amount <= 0:
26         return(0)
27
28     # Get user balance
29     balance = self.balances[msg.sender].balance
```



# Remix : les alternatives



The screenshot shows the Visual Studio Code Marketplace page for the 'solidity' extension. At the top, there's a navigation bar with 'Visual Studio | Marketplace' on the left, 'Sign in' and 'Search extensions' on the right, and a breadcrumb trail 'Visual Studio Code > Programming Languages > solidity'. Below the header, the extension card for 'solidity' is displayed. It features a large Ethereum logo icon, the name 'solidity' in bold, the author 'Juan Blanco', install count '113,152 installs', download count '475,699 downloads', a 5-star rating '(11)', and a 'Free' status. A brief description 'Ethereum Solidity Language for Visual Studio Code' is shown below the title. Two buttons are present: a green 'Install' button and a link 'Trouble Installing?'. At the bottom of the card, there are tabs for 'Overview' (which is selected), 'Q & A', and 'Rating & Review'.

Overview

Q & A

Rating & Review

## Solidity support for Visual Studio code

Solidity is the language used in Ethereum to create smart contracts, this extension provides:

- Syntax highlighting
- Snippets
- Compilation of the current contract (Press F1 Solidity : Compile Current Solidity Contract), or F5
- Compilation of all the contracts (Press F1 Solidity : Compile all Solidity Contracts), or Ctrl+F5 / Cmd+F5
- Code completion for all contracts / libraries in the current file and all referenced imports
- Default project structure (solidity files needs to be in the 'src' directory, and libraries in the 'lib' directory)

## Categories

Programming Languages Snippets

## Tags

blockchain compiler ethereum keybindings  
snippet solhint solidity

## Resources



## Atelier 2 : Développer votre propre monnaie virtuelle dans Remix



# Atelier 1 : Créer votre crypto-monnaie

---

- ▶ **Etape 1**
  - ▶ Ecrire votre smart contrat
  - ▶ <https://github.com/hakiri/Tuto-Blockchain/blob/master/erc20-tutorial.sol>
  - ▶ Dans Remix (Fichier Token.sol ou
- ▶ **Etape 2:**
  - ▶ Compiler le smart contrat sur Remix
- ▶ **Etape 3:**
  - ▶ Exécuter et debugger votre smart contrat
- ▶ **Etape 4:**
  - ▶ Déployer le smart contrat dans la blockchain,
  - ▶ et éventuellement créer votre propre entreprise décentralisée
- ▶ **Utiliser le smart contract disponible sur github**
  - ▶ <https://github.com/hakiri/Tuto-Blockchain>
  - ▶ <https://github.com/hakiri/Tuto-Blockchain.git>



## Atelier 2 : Développer votre propre monnaie virtuelle avec Truffle



Niveau avancé

# Exemple 2 : Construisons une crypto-monnaie

---

- ▶ Configuration de l'environnement de développement :
  - ▶ Télécharger et installer Nodejs, ici <https://nodejs.org/en/download/>
  - ▶ Télécharger et installer ganache (CLI et GUI) ici <https://truffleframework.com/ganache>
  - ▶ Installer le Framework truffle avec le gestionnaire de paquets npm:
    - ▶ `npm install truffle -g`
  - ▶ Un IDE ou éditeur de texte (recommandation:Visual Studio Code)
- ▶ Rédigez votre premier contrat intelligent dans Truffle en utilisant Solidity:
  - ▶ Documents Truffle –
    - ▶ <https://truffleframework.com/docs/truffle/overview>
  - ▶ Guide de solidité –
    - ▶ <https://solidity.readthedocs.io/fr/v0.4.25/>
  - ▶ Pour développer en JavaScript, le guide web3.js:
    - ▶ <http://web3js.readthedocs.io/fr/1.0/web3-eth.html>
  - ▶ Pour développer en Python, le guide web3.py –
    - ▶ <https://web3py.readthedocs.io/en/stable/>

# Introduction au Framework Truffle

---

- ▶ Les outils en ligne de commande comme « geth » et « solc » permettent d'interagir directement avec le smart contrat
  - ▶ On peut utiliser des scripts « Bash » pour automatiser cette interaction, compiler, puis déployer le contrat...
  - ▶ Cette approche est toujours assez rudimentaire (expérience sous-optimale du script Bash).
- ▶ Truffle est apparaît pour résoudre ces problèmes rudimentaires
  - ▶ permettre une meilleur automatisation, lorsqu'on compile et déploie un smart contrat
  - ▶ Il permet aussi de développer des applications décentralisées avancées

# Introduction au Framework Truffle

---

- ▶ Truffle permet de répondre à de nombreux problèmes rencontrés dans le développement d'applications blockchain
  - ▶ Compilation :
    - ▶ plusieurs versions du compilateur « solc » doivent être prises en charge simultanément, avec une indication claire de celle qui est utilisée.
  - ▶ Environnements
    - ▶ Les contrats doivent comporter des environnements de développement, d'intégration et de production, chacun avec sa propre adresse de nœud Ethereum, ses comptes, etc.
  - ▶ Test
    - ▶ Les contrats doivent être testables avant leur déploiement

# Introduction au Framework Truffle

---

- ▶ Configuration
  - ▶ Vos environnements de développement, d'intégration et de production doivent être encapsulés dans un fichier de configuration pour pouvoir être validés par git et réutilisés par vos coéquipiers.
- ▶ Intégration Web3js
  - ▶ Web3.js est un Framework JavaScript permettant une communication plus facile avec des smart contrats intelligents à partir d'applications Web.
  - ▶ Truffle va plus loin et active l'interface Web3.js à partir de la console Truffle.
  - ▶ Vous pouvez ainsi appeler des fonctions Web tout en restant en mode de développement, en dehors du navigateur.

# Installation de Truffle

---

## ▶ Installation

- ▶ La meilleure façon d'installer Truffle consiste à utiliser le Node Package Manager (npm).
- ▶ Après avoir configuré NPM sur votre ordinateur, installez Truffle en ouvrant le terminal et en tapant ceci : **npm install -g truffle**

## ▶ Initialiser un nouveau projet MetaCoin

- ▶ Ouvrez un Terminal et positionnez-vous dans le dossier où vous souhaitez initialiser le projet.
  - ▶ **mkdir MetaCoin**
  - ▶ **cd MetaCoin**
  - ▶ **truffle unbox metacoin**

# Structure du projet Truffle

```
C:\Users\hakir\Desktop\Tuto Blockchain\metacoin>truffle unbox metacoin
```

- ✓ Preparing to download
- ✓ Downloading
- ✓ Cleaning up temporary files
- ✓ Setting up box

Unbox successful. Sweet!

Commands:

Compile contracts: truffle compile

Migrate contracts: truffle migrate

Test contracts: truffle test

```
C:\Users\hakir\Desktop\Tuto Blockchain\metacoin>tree /F
Structure du dossier
Le numéro de série du volume est A0D6-90FC
C:.
    LICENSE
    truffle-config.js

    contracts
        .placeholder
        ConvertLib.sol
        MetaCoin.sol
        Migrations.sol

    migrations
        1_initial_migration.js
        2_deploy_contracts.js

    test
        .placeholder
        metacoin.js
        TestMetacoin.sol
```

# Structure du projet

---

- ▶ **Le dossier contracts**
  - ▶ Comprend l'ensemble de fichiers **solidity** pour votre smart contrat
  - ▶ Vous pouvez modifier, ajouter, supprimer et mettre à jour vos contrats
  - ▶ un fichier « `Migrations.sol` », est un fichier spécial pour automatiser le déploiement de vos contrats
  - ▶ Lorsque Truffle compile le projet, il passe par le dossier des contrats et compile tous les fichiers compatibles.
- ▶ **Le dossier migrations**
  - ▶ Contient les fichiers nécessaire pour déployer vos smart contrats dans la blockchain
  - ▶ Vous pouvez ouvrir les fichiers dans votre IDE (VS Code) et voir les appels de fichiers (importation de fichiers)
- ▶ **Le dossier test**
  - ▶ Pour le test unitaire et le test logiciels
- ▶ **Le fichier de configuration s'appelle `truffle.js` ou `truffle-config.js`.**
- ▶ **Environments:** Develop, TestNet, Live

# Compilation du projet

- ▶ Compilation
  - ▶ truffle compile

```
C:\Users\hakir\Desktop\Tuto Blockchain\metacoin>truffle compile

Compiling your contracts...
=====
> Compiling .\contracts\ConvertLib.sol
> Compiling .\contracts\MetaCoin.sol
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\ConvertLib.sol
> Artifacts written to C:\Users\hakir\Desktop\Tuto Blockchain\metacoin\build\contracts
> Compiled successfully using:
  - solc: 0.5.0+commit.1d4f565a.Emscripten clang
```

# Déploiement dans le blockchain locale

- ▶ Migration sur localhost
  - ▶ truffle migrate
  - ▶ NB: vous pouvez aussi spécifier la migration dans la blockchain de production (MainNet)
    - ▶ truffle migrate --network live

```
C:\Users\hakir\Desktop\Tuto Blockchain\metacoin>truffle migrate
Compiling your contracts...
=====
> Compiling .\contracts\ConvertLib.sol
> Artifacts written to C:\Users\hakir\Desktop\Tuto Blockchain\metacoin\build\contracts
> Compiled successfully using:
  - solc: 0.5.0+commit.1d4f565a.Emscripten.clang

Could not connect to your Ethereum client with the following parameters:
  - host      > 127.0.0.1
  - port      > 7545
  - network_id > 5777
Please check that your Ethereum client:
  - is running
  - is accepting RPC connections (i.e., "--rpc" option is used in geth)
  - is accessible over the network
  - is properly configured in your Truffle configuration file (truffle-config.js)

Truffle v5.0.10 (core: 5.0.10)
Node v11.11.0
```

# Test d'intégration

- ▶ **Test**
  - ▶ Afin de tester vos contrats, lancez ceci:
    - ▶ truffle test
  - ▶ Ou vous pouvez lancer un test spécifique en lançant ceci:
    - ▶ truffle test  
./path/to/FileTest.sol

```
C:\Users\hakir\Desktop\Tuto Blockchain\metacoin>truffle test
Using network 'test'.


Compiling your contracts...
=====
> Compiling .\contracts\ConvertLib.sol
> Compiling .\test\TestMetacoin.sol
> Artifacts written to C:\Users\hakir\AppData\Local\Temp\test-11933-56404-1974rh.61ttm
> Compiled successfully using:
  - solc: 0.5.0+commit.1d4f565a.Emscripten.clang


TestMetacoin
  ✓ testInitialBalanceUsingDeployedContract (109ms)
  ✓ testInitialBalanceWithNewMetaCoin (112ms)

Contract: MetaCoin
  ✓ should put 10000 MetaCoin in the first account (117ms)
  ✓ should call a function that depends on a linked library (71ms)
  ✓ should send coin correctly (249ms)


  5 passing (12s)
```

# Résultat de la compilation

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Debug, Terminal, Help.
- Explorer:** Shows the project structure:
  - OPEN EDITORS: MetaCoin.sol contracts (9+), Migrations.sol contracts (9+), 1\_initial\_migration.js migrations (6), truffle-config.js (2).
  - METACOIN: build\contracts (ConvertLib.json, MetaCoin.json, Migrations.json), contracts (ConvertLib.json), migrations, test, LICENSE, truffle-config.js.
- Code Editor:** The active file is ConvertLib.json, which contains Solidity code for a contract named ConvertLib. The code defines a function convert with parameters amount (uint256) and conversionRate (uint256), returning convertedAmount (uint256). It is marked as payable: false, stateMutability: pure, and type: function. The metadata includes compiler version 0.5.0+commit.1d4f565a, language Solidity, and bytecode.
- Bottom Status Bar:** PROBLEMS 220, OUTPUT, DEBUG CONSOLE, TERMINAL, 1: cmd, Ln 441, Col 35, Spaces: 2, UTF-8, LF, JSON, ESLint, and icons for Docker Hub, Azure Container Registry, Docker Containers, Docker Images, and Suggested Docker Hub Images.
- Bottom Progress Bar:** 191 ▲ 29 Installing packages...

## Outils complémentaires: OpenZeppelin

---

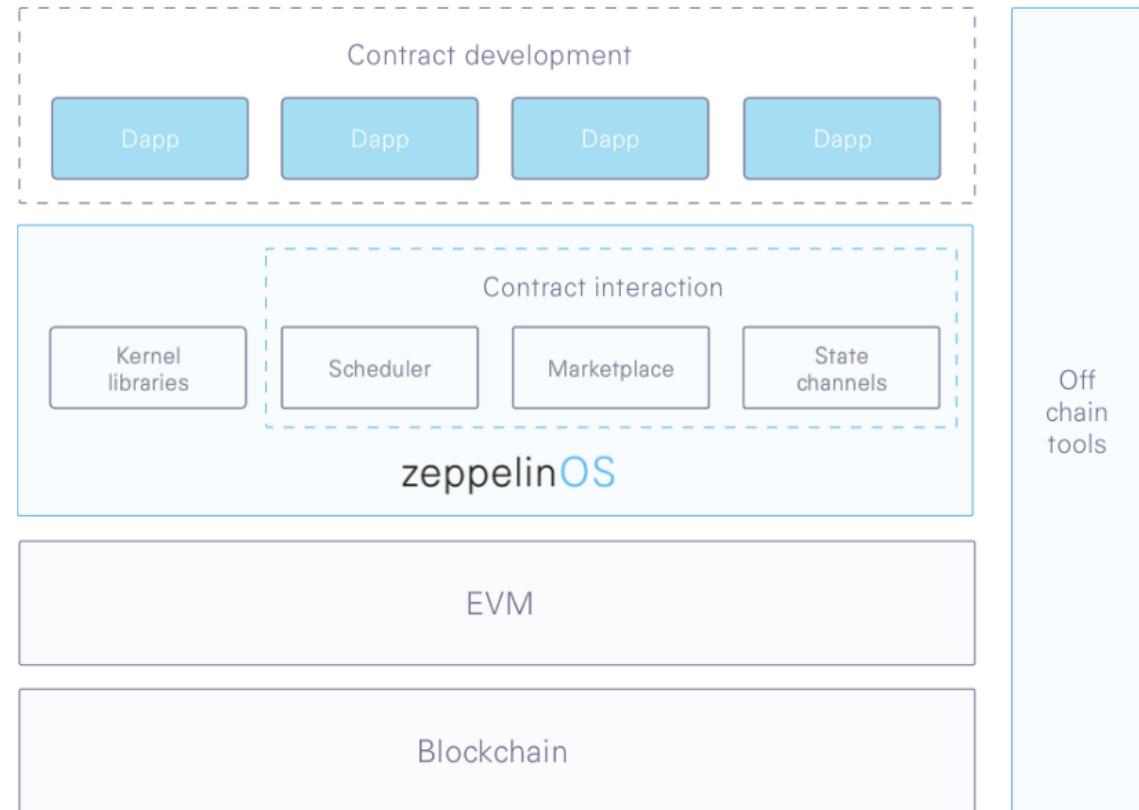
- ▶ Bibliothèque OpenZeppelin <https://openzeppelin.org/>
  - ▶ C'est un Framework composé de nombreux modèles de code Solidity et de modules de contrat intelligents, écrits de manière sécurisée.
  - ▶ Un tutoriel sur l'utilisation d'OpenZeppelin avec Truffle et Ganache.
    - ▶ <https://truffleframework.com/tutorials/robust-smart-contracts-with-openzeppelin>
    - ▶ contient un ensemble de contrats de publication de jetons sur Ethereum - pour les jetons ERC20, y compris un contrat BasicToken, BurnableToken, CappedToken.

# ZeppelinOs

- ▶ **Middleware ZeppelinOs**

<https://zeppelinos.org/>

- ▶ C'est plateforme d'outils et de services décentralisés libre s'ajoutant à l'EVM, pour développer et gérer en toute sécurité des applications de contrat intelligentes.
- ▶ une description technique de ce framework se trouve ici
  - ▶ <https://blog.zeppelin.solutions/technical-details-of-zeppelinos-d3cf4da591f7>



# Meilleures pratiques de sécurité des contrats Ethereum Smart

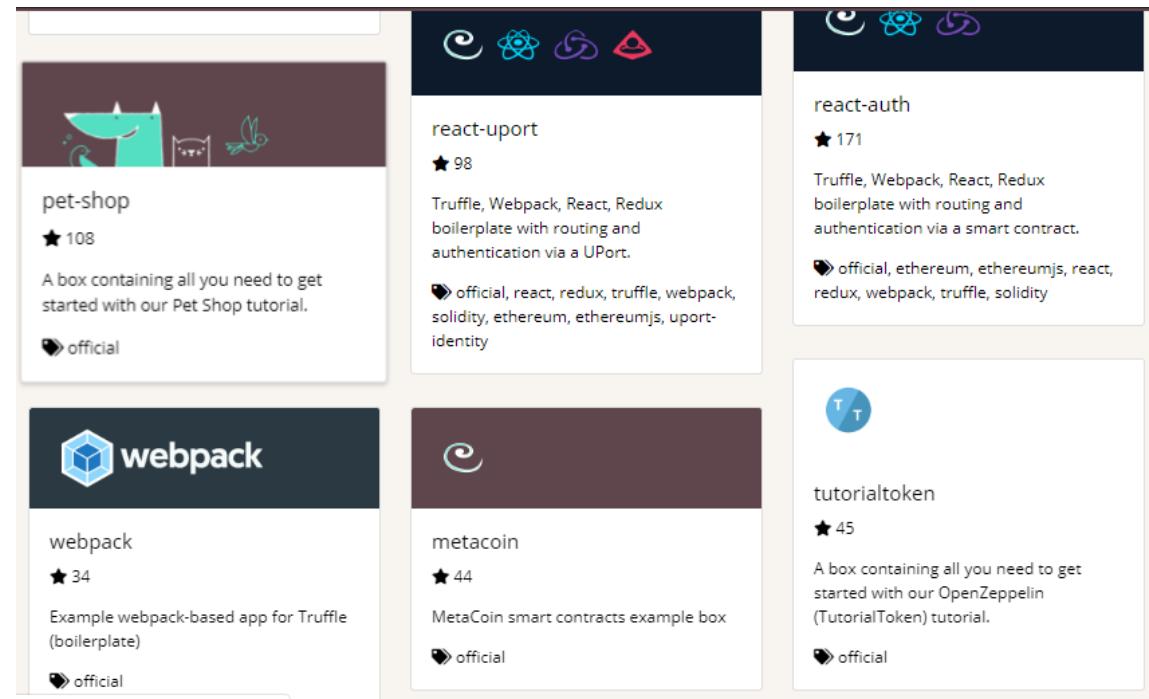
---

- ▶ **ConsenSys** <https://consensys.net/>
  - ▶ une entreprise suisse d'investissements dans l'espace blockchain décentralisé, notamment l'espace Ethereum.
  - ▶ ils ont rassemblé un joli petit recueil de bonnes pratiques pour les contrats intelligents Ethereum. B
  - ▶ Bien qu'il ne s'agisse pas d'un code, il contient un bon nombre de bons et de mauvais exemples de code Solidity.
- ▶ **TokenMarket** <https://tokenmarket.net/>
  - ▶ société qui a publié un référentiel de contrats Solidity et d'outils de gestion des ventes de jetons.
  - ▶ Permet d'utiliser ou de développer les contrats OpenZeppelin existants, les qualifiant de «gold standard» des contrats Solidity.
  - ▶ une grande partie du code est basée sur OpenZeppelin, et elle s'enrichit ensuite

# Outils complémentaires: Truffle Boxes

- ▶ Truffle inclut quelques boites comprenant du code en JavaScripts, Solidity, Workflow, etc. pour vous aider à développer des DApps
  - ▶ la boîte standard du projet Webpack
  - ▶ Drizzle: ensemble de bibliothèques frontales basées sur React/Redux qui pour créer frontend DApps.
  - ▶ React box: une application simple pour interagir avec des contrats intelligents à partir d'une application frontale React. (**truffle unbox react**)

<https://truffleframework.com/boxes>



# Petite démonstration : installer et tester React package

---

- ▶ `npx truffle unbox react`
- ▶ Ou bien :
  - ▶ `npm install -g truffle`
  - ▶ `truffle unbox react`
- ▶ Ouvrir une console de développement
  - ▶ `truffle develop`
- ▶ Compiler et migrer les contrats intelligents
  - ▶ `truffle(develop)> compile`
  - ▶ `truffle(develop)> migrate`
- ▶ Un dossier client est créé après la migration
- ▶ Tester vos smart contracts dans une nouvelle console et changer le dossier de votre react-pack
  - ▶ `cd client`
  - ▶ `npm run start`
- ▶ Dans un nouveau terminal
  - ▶ `Truffle test`
- ▶ Ou bien dans la console de développement :
  - ▶ `test`
- ▶ Une nouvelle page web s'ouvre sur  
<http://localhost:3000/>

# Travail personnel

---

- ▶ Travail à faire : installer et tester Web package
- ▶ Créer un dossier vide et taper dedans:
  - ▶ truffle unbox webpack
  - ▶ truffle compile
  - ▶ truffle migrate
  - ▶ Ganache-cli
  - ▶ cd ./app
  - ▶ npm run dev
  - ▶ <http://127.0.0.1:8080>

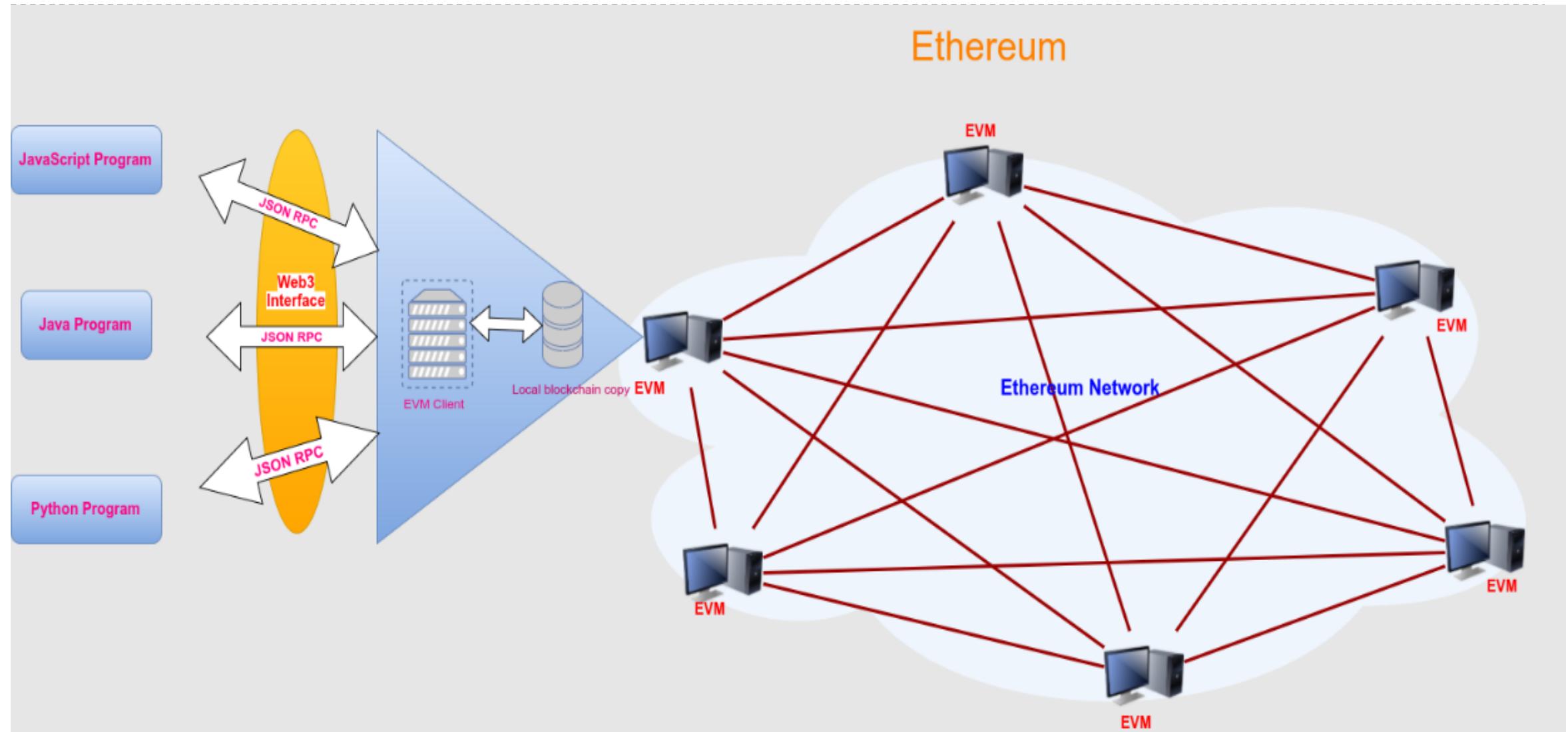
# Travail personnel

---

- ▶ Copier le code source du projet
  - ▶ git clone <https://github.com/ajb413/crowdsalable-eth-token.git>
  - ▶ cd crowdsalable-eth-token
  - ▶ npm i
  - ▶ truffle develop
- ▶ ## Démarrer la blockchain et la console de développement truffle
  - ▶ truffle(develop)> compile
  - ▶ truffle(develop)> migrate
  - ▶ truffle(develop)> test
- ▶ Dans une deuxième fenêtre de ligne de commande, accédez au répertoire du projet.
  - ▶ npm run dev
  - ▶ Project is running at http://localhost:8080/
- ▶ Accédez à http://localhost:8080/ dans un navigateur Web

## Atelier 3: Développer une application Blockchain décentralisée avec Python

# Interagir avec la Blockchain



# Préparer l'environnement de développement

---

## ▶ Nous allons utiliser une machine virtuelle en ligne

- ▶ Créer un compte <https://labs.play-with-docker.com>
- ▶ Créer un compte <https://infura.io/> et créer un projet dans infura.io
- ▶ Créer une nouvelle instance de machine virtuelle Docker

## ▶ Tapez dans le terminal :

- ▶ \$ docker container run -it ubuntu
- ▶ # apt update
- ▶ # apt-get install -y nano virtualenv python-pip python3-pip python3-venv python3.6-dev

## ▶ Configurer l'environnement virtuel python :

- ▶ root@045efa6dd3b1:# pip install virtualenv
- ▶ root@045efa6dd3b1:# virtualenv venv
- ▶ root@045efa6dd3b1:# virtualenv -p /usr/bin/python3.6 venv
- ▶ root@045efa6dd3b1:# source venv/bin/activate
- ▶ (venv) root@045efa6dd3b1:# pip install web3

Sous Ubuntu:

\$python3 -m venv venv  
\$ . venv/bin/activate  
\$ pip install web3

# Tester l'API Web3py dans un terminal

- ▶ Dans votre terminal, tapez :# python

```
>>> from web3 import Web3, HTTPProvider

>>> web3 = Web3(HTTPProvider('http://localhost:8545'))

>>> web3
<web3.main.Web3 object at 0x7f7c38b0dcf8>

>>> Web3.toWei(1, 'ether')
1000000000000000000000000

>>> web3.fromWei(1000000000000000000000000, 'ether')
Decimal('1')

>>> from web3.auto import w3
>>> existing_filter = web3.eth.filter(filter_id="0x0")
>>> existing_filter
<web3.utils.filters.LogFilter object at 0x7f7c371e8048>
```

# Tester l'API Web3py dans un terminal

---

- ▶ Se connecter dans votre compte infura et créer un nouveau projet vide
- ▶ Copier le lien “ENDPOINT” à votre projet pour l'utiliser
- ▶ Dans votre terminal Docker, tapez :# python
- ▶ Lancer la série de commande suivantes:

```
>>> from web3 import Web3, HTTPProvider
>>> infura_url = 'https://mainnet.infura.io/v3/00919abdfabf4cecae37fe653ae1a9a1'
>>> web3 = Web3(HTTPProvider(infura_url))
>>> web3.isConnected()
True
>>> web3.eth.blockNumber
7578773
>>> web3.eth.blockNumber
7578773
```

# Le premier test est réussi

The screenshot shows a cloud management interface with a sidebar and a main session details view.

**Header:** 03:52:18

**Buttons:** CLOSE SESSION

**Instances:** Instances + ADD NEW INSTANCE

**Session Details:**

- IP:** 192.168.0.8
- Memory:** 21.17% (846.7MiB / 3.906GiB)
- CPU:** 1.06%
- SSH:** ssh ip172-18-0-26-bivjddvr8gag00f9jm20@direct.labs.pl

**Actions:** DELETE

**Terminal Output:**

```
(venv) root@883c145879a5:/# python
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from web3 import Web3, HTTPProvider
>>> infura_url = 'https://mainnet.infura.io/v3/7842b8c809dd48e5bdb0b85243bee579'
>>> web3 = Web3(HTTPProvider(infura_url))
>>> web3.isConnected()
True
>>> web3.eth.blockNumber
7624799
>>> web3.eth.blockNumber
7624799
>>> 
```

# Tutorial 1

---

- ▶ On va écrire un script python pour interagir avec Ethereum blockchain
- ▶ Ecrire un script python app.py, par exemple:
  - ▶ nano app.py
  - ▶ Copier le code source (slide suivant) dans le script app.py
- ▶ pour exécuter votre programme
  - ▶ **\$ python app.py**

# Le script de test

---

```
1 from web3 import Web3
2 # the url is the link to my project in my profile on infura
3 # this need you to create a url to your own project on infura
4 infura_url_ropsten = 'https://ropsten.infura.io/v3/00919abdfabf4cecae37fe653ae1a9a1'
5 web3 = Web3(HTTPProvider(infura_url_ropsten))
6
7 web3.isConnected()
8 print(web3.eth.blockNumber)
9 # account 1 is my own account on metamask on ropsten
10 account1 = "0x80e9B8f891c326bc8c25dA59672DC5F363C4bC9A"
11 balance = web3.eth.getBalance(account1)
12 print(balance)
13
14 web3.fromWei(balance, 'ether')
15 print(web3.fromWei(balance, 'ether'))
16
```

## Tutorial 2

---

- ▶ Écrire un script python pour envoyer de la cryptomonnaie dans la blockchain avec Web3.py
- ▶ **Ouvrir <https://etherscan.io> et chercher le jeton (Token) OmiseGO dans le lien suivant**  
<https://etherscan.io/token/0xd26114cd6EE289AccF82350c8d8487fedB8A0C07>
- ▶ Vous pouvez remarquer le smart contract de ce Token, cliquer sur l'adresse et aller jusqu'au code du smart contract :

# Script Python (Partie 1)

```
from web3 import Web3
import json

# the url is the link to my project in my profile on infura
# this need you to create a url to your own project on infura
infura_url_mainnet = 'https://mainnet.infura.io/v3/00919abdfabf4cecae37fe653ae1a9a1'
web3 = Web3(HTTPProvider(infura_url_mainnet))
print(web3.isConnected()) # you should see true

# ABI of the smart contract
abi = json.loads('[{"constant":true,"inputs":[],"name":"mintingFinished","outputs":[{"name":"","type':
#address of the smart contract
address = "0xd26114cd6EE289AccF82350c8d8487fedB8A0C07"

contract = web3.eth.contract(address=address,abi=abi)

print(contract)
```

# Adresse du smart contract à ajouter à votre script

## Profile Summary



Contract: 0xd26114cd6EE289AccF82350c8d8487fedB8A0C07

Decimals: 18

Official Site: <https://omisego.network/>

Social Profiles:



# Smart contract : code source disponible sur le site etherscan

Transactions Internal Txns Erc20 Token Txns **Code** Read Contract Write Contract Events Analytics New Comments

**⚠ Warning:** The compiled contract might be susceptible to ExpExponentCleanup (medium/high-severity), NestedArrayFunctionCallDecoder (medium-severity), ZeroFunctionSelector (very low-severity), DelegateCallReturnValue (low-severity), ECRecoverMalformedInput (medium-severity), SkipEmptyStringLiteral (low-severity) Solidity Compiler Bugs.

✓ Contract Source Code Verified (Exact Match)

Contract Name:	OMGToken	Optimization Enabled:	Yes
Compiler Version	v0.4.11+commit.68ef5810	Runs (Optimizer):	200

Contract Source Code Find Similar Contracts Copy Open in new tab

```
1 pragma solidity ^0.4.11;
2
3
4 /**
5  * Math operations with safety checks
6  */
7 library SafeMath {
8     function mul(uint a, uint b) internal returns (uint) {
9         uint c = a * b;
10        assert(a == 0 || c / a == b);
11        return c;
12    }
13}
```

# ABI du smart contract à copier et ajouter au script

```
18     return c;
19 }
20
21 /*
22  * function sub(uint a, uint b) internal returns (uint) {
23  *     assert(b <= a);
24  *     return a - b;
25  */
```

Contract ABI

Export ABI ▾

```
[{"constant":true,"inputs":[],"name":"mintingFinished","outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"}, {"constant":true,"inputs":[],"name":"name","outputs":[{"name":"","type":"string"}],"payable":false,"type":"function"}, {"constant":false,"inputs":[{"name":"_spender","type":"address"}],"name":"_value","type":"uint256"}, {"name":"approve","outputs":[],"payable":false,"type":"function"}, {"constant":true,"inputs":[],"name":"totalSupply","outputs":[{"name":"","type":"uint256"}],"payable":false,"type":"function"}, {"constant":false,"inputs":[{"name":"_from","type":"address"}, {"name":"_to","type":"address"}, {"name":"_value","type":"uint256"}],"name":"transferFrom","outputs":[],"payable":false,"type":"function"}, {"constant":true,"inputs":[],"name":"decimals","outputs":[{"name":"","type":"uint256"}],"payable":false,"type":"function"}, {"constant":false,"inputs":[],"name":"unpause","outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"}, {"constant":false,"inputs":[{"name":"_to","type":"address"}, {"name":"_amount","type":"uint256"}],"name":"mint","outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"}, {"constant":true,"inputs":[],"name":"paused","outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"}, {"constant":true,"inputs":[{"name":"_owner","type":"address"}],"name":"balanceOf","outputs":[{"name":"balance","type":"uint256"}],"payable":false,"type":"function"}, {"constant":false,"inputs":[],"name":"finishMinting","outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"}, {"constant":false,"inputs":[],"name":"pause","outputs":[]}]
```

This website uses cookies to improve your experience and has an updated [Privacy Policy](#).

Got It

## Script Python (Partie 2)

```
# interagir avec le smart contract
totalSupply = contract.functions.totalSupply().call()
print(contract.functions.totalSupply().call())

# print the totalSupply (th same)
print(totalSupply)
print(web3.fromWei(totalSupply, 'ether'))

# call functions
print(contract.functions.name().call())
print(contract.functions.symbol().call())

holderAddress = '0x2551d2357c8da54b7d330917e0e769d33f1f5b93'
valid_address = web3.toChecksumAddress(holderAddress)
balance = contract.functions.balanceOf(valid_address).call()
print(web3.fromWei(balance, 'ether'))
```

# Liste des fonctions du smart contract disponible, à utiliser dans le script python

Transactions Internal Txns Erc20 Token Txns Code  Read Contract Write Contract Events Analytics  Now Comments

---

 Read Contract Information [Reset]

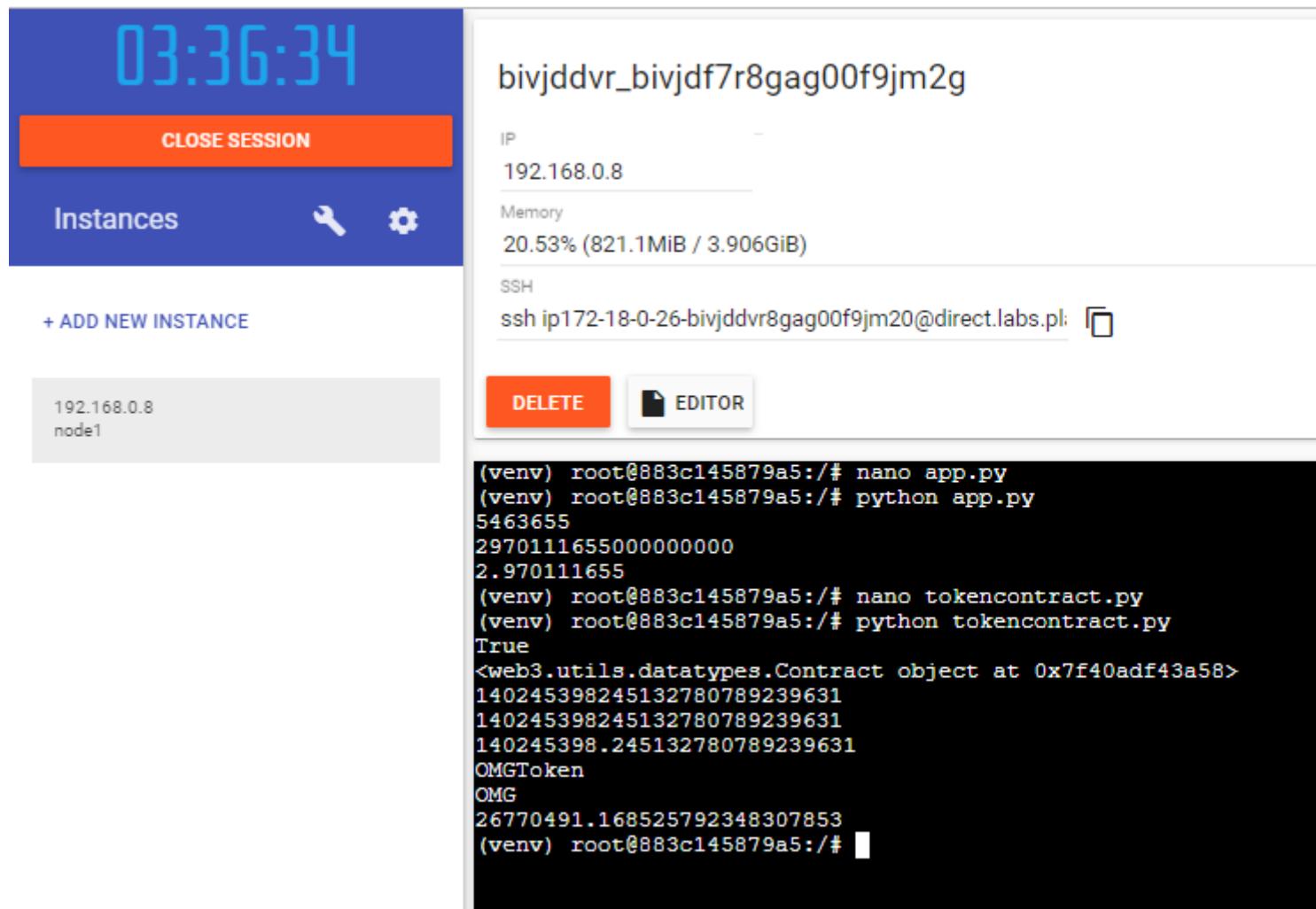
1. mintingFinished	True <i>bool</i>
2. name	OMGToken <i>string</i>
3. totalSupply	140245398245132780789239631 <i>uint256</i>
4. decimals	18 <i>uint256</i>

# Liste des utilisateurs (holders) qui utilisent le Token

on souhaite afficher leurs balances dans le script (méthode balanceOf(address du holder))

Transfers	Holders	Info	Chart	Exchange	DEX Trades	Read Contract	Write Contract	Analytics <small>New</small>	Comments	
Token Holders Chart										
Top 1,000 holders (From a total of 652,667 holders)										
First < Page 1 of 20 > Last										
Rank	Address						Quantity		Percentage	
1	0x2551d2357c8da54b7d330917e0e769d33f1f5b93						26,770,491.1685258		19.0883%	
2	0x4e9ce36e442e55ecd9025b9a6e0d88485d628a67	(Binance 6)					10,910,067.7554955		7.7793%	
3	0x66f820a414680b5bcda5eeeca5dea238543f42054	(Bittrex 3)					10,026,364.3165323		7.1492%	
4	0xadb2b42f6bd96f5c65920b9ac88619dce4166f94	(Huobi 7)					4,918,539.72296049		3.5071%	

# Exécuter votre script



# Tutorial 3 (Niveau avancé)

---

- ▶ Manipulation de smart contract dans une blockchain locale personnelle avec ganache
- ▶ Télécharger et installer ganache : <https://truffleframework.com/ganache>
- ▶ Sous Windows :
  - ▶ Télécharger VSCode et Visual Studio 2019 ou 2017 (avec C++ build tools, python dev tools)
  - ▶ Activer virtualenv (`c:\>pip install virtualenv && virtualenv venv`)
- ▶ Sous Linux :
  - ▶ télécharger les packages python en dessus et activer virtualenv

# Démonstration

---

- ▶ J'utilise VS Code sur machine avec les outils préconfiguré
- ▶ Les fichiers sont disponibles sur mon compte github
  - ▶ <https://github.com/hakiri/Tuto-Blockchain>
- ▶ Si vous utiliser Windows, et vous souhaitez faire l'exemple, il faut soit
  - ▶ installer une machine virtuelle sous VirtualBox ou VM Player (solution préférée pour débuter)
  - ▶ Soit installer les dépendances sous Windows

# Script 1/2

```
1 from web3 import Web3
2 import json
3
4 # url to ganache blockchain
5 ganache_url = "http://127.0.0.1:7545"
6
7 # connect to local blockchain
8 web3 = Web3(HTTPProvider(ganache_url))
9
10 #Check if it is connected (should see true)
11 print(web3.isConnected()) # should see true
12 print(web3.eth.blockNumber) # should see 0
13
14 # Need two ganache accounts
15 account_1 = "0x336128282fb144d8CedcB6B053c2A2e161d040c8"
16 account_2 = "0xB51b607A224060027572D7Ef3dff5cE59ef6ab34"
17
18 #First account Private Key from ganache account to send crypto currency
19 # To Allow sending Ether from account 1 to account 2
20 private_key = "d56211d872791ce8874a7800fde1ddd46aeabdb61b0f507ba17e7a08dab3f81"
21
22 # Get the transaction nonce
23 nonce = web3.eth.getTransactionCount(account_1)
24
```

# Script 2/2

```
25 # Build a transaction :
26 # A dictionary contains all the transactions information
27 ▼ tx = {
28     'nonce':nonce,
29     'to': account_2, # Account we will a transaction to -- From account will be signed
30     'value': web3.toWei(1,'ether'), # Actual amount of Ether will send to account_2
31     'gas': 2000000, # gas limit to pay the transaction fees when sending it -- needed by
32     'gasPrice': web3.toWei('50','gwei'), #
33 }
34 # Need to sign a transaction before sending
35 signed_tx = web3.eth.account.signTransaction(tx,private_key)
36 # Send transaction
37 # web3.eth.sendRawTransaction(signed_tx.rawTransaction)
38 # Send signed transaction and get tx_hash
39 tx_hash = web3.eth.sendRawTransaction(signed_tx.rawTransaction)
40 # Get transaction hash
41 print(tx_hash)
42
43 # convert tx_hash to hexadecimal format
44 # As you execute the python script you send 1 Ether from account_1 to account_2
45 # You can see the results on Ganache, too !!
46
47 print(web3.toHex(tx_hash))
48 print("see Accounts on Ganache GUI \n")
49 print("see Transactions on Ganache GUI \n")
```

# Interface de Ganache (Blockchain locale)

The screenshot shows the Ganache interface with the following details:

ADDRESS	BALANCE	TX COUNT	INDEX	KEY
0x336128282fb144d8CedcB6B053c2A2e161d040c8	97.00 ETH	3	0	🔑
0xB51b607A224060027572D7Ef3dff5cE59ef6ab34	103.00 ETH	0	1	🔑
0x19Be7d3127A3f6C4888aCbcB3B3c9EdC675061f2	100.00 ETH	0	2	🔑
0x6d9D130F975C383aA9f9505815c67d05A6E9e6aA	100.00 ETH	0	3	🔑
0x06e3a08cBa09312Ea0235536cD28374D5ab4D900	100.00 ETH	0	4	🔑
0x5619eE244BF5194238997CCe097184620f2D0871	100.00 ETH	0	5	🔑
ADDRESS	BALANCE	TX COUNT	INDEX	

# Les transactions dans la blockchain locale

The screenshot shows the Ganache interface with the following details:

TX HASH		CONTRACT CALL	
0x48239a0eea85767de8f8e11a3127b8026745e0a8db44d7450617cc54c525d41a			
FROM ADDRESS 0x336128282fb144d8CedcB6B053c2A2e161d040c8	TO CONTRACT ADDRESS 0xB51b607A224060027572D7Ef3dff5cE59ef6ab34	GAS USED 21000	VALUE 1000000000000000000000000
TX HASH		CONTRACT CALL	
0xe014d0dc57c52c5923152350a076a8e58cd0ef41e242fd2b9493d6b6261a44cd			
FROM ADDRESS 0x336128282fb144d8CedcB6B053c2A2e161d040c8	TO CONTRACT ADDRESS 0xB51b607A224060027572D7Ef3dff5cE59ef6ab34	GAS USED 21000	VALUE 1000000000000000000000000
TX HASH		CONTRACT CALL	
0xb2086164740bc18973c0937372d2a3f2b603acf3775e06ab77738d30f2b1520a			
FROM ADDRESS 0x336128282fb144d8CedcB6B053c2A2e161d040c8	TO CONTRACT ADDRESS 0xB51b607A224060027572D7Ef3dff5cE59ef6ab34	GAS USED 21000	VALUE 1000000000000000000000000

# La dernière transaction

The screenshot shows the Ganache interface with the following details:

- Header:** ACCOUNTS, BLOCKS, TRANSACTIONS (selected), CONTRACTS, EVENTS, LOGS, UPDATE AVAILABLE, SEARCH FOR BLOCK NUMBERS OR TX HASHES.
- Network Status:** CURRENT BLOCK 3, GAS PRICE 200000000000, GAS LIMIT 6721975, HARDFORK PETERSBURG, NETWORK ID 5777, RPC SERVER HTTP://127.0.0.1:7545, MINING STATUS AUTOMINING, WORKSPACE QUICKSTART.
- Transaction Details:** TX 0x48239a0eea85767de8f8e11a3127b8026745e0a8db44d7450617cc54c525d41a. The transaction originated from address 0x336128282fb144d8CedcB6B053c2A2e161d040c8 and was sent to contract address 0xB51b607A224060027572D7Ef3dff5cE59ef6ab34.
- Transaction Parameters:** VALUE 1.00 ETH, GAS USED 21000, GAS PRICE 50000000000, GAS LIMIT 2000000, MINED IN BLOCK 3.
- TX Data:** 0x.
- Events:** NO EVENTS.

# Les blocks

- Voir tous les blocs (04 blocs) utilisés pour envoyer les transactions

The screenshot shows the Ganache interface with the title "Ganache". The top navigation bar includes icons for Accounts, Blocks (highlighted in orange), Transactions, Contracts, Events, Logs, and an "UPDATE AVAILABLE" notification. A search bar at the top right allows searching for block numbers or tx hashes.

The main content area displays four blocks:

BLOCK	MINED ON	GAS USED	TRANSACTIONS
3	2019-04-16 17:23:21	21000	1 TRANSACTION
2	2019-04-16 17:20:39	21000	1 TRANSACTION
1	2019-04-16 17:19:06	21000	1 TRANSACTION
0	2019-04-16 16:22:38	0	NO TRANSACTIONS

Below the blocks, the interface shows various configuration parameters: CURRENT BLOCK (3), GAS PRICE (20000000000), GAS LIMIT (6721975), HARDFORK (PETERSBURG), NETWORK ID (5777), RPC SERVER (HTTP://127.0.0.1:7545), MINING STATUS (AUTOMINING), and WORKSPACE (QUICKSTART). There are also "SAVE", "SWITCH", and "SETTINGS" buttons.



## Atelier 4: Application de Vote



# Travail à faire : Utilisation de Remix

---

- ▶ Aller au dépôt git : <https://github.com/hakiri/Tuto-Blockchain>
- ▶ Utiliser le fichier Election.sol ou ElectionI.sol
- ▶ Compiler et exécuter
- ▶ Dépendances : aucune
- ▶ Comparer le contrat Election.sol et ElectionI.sol

# Travail à faire : Utilisation de Truffle

---

▶ **Indications: Suivre les instructions suivantes**

- ▶ npm install -g truffle
- ▶ git clone https://github.com/tko22/truffle-webpack-boilerplate.git
- ▶ cd truffle-webpack-boilerplate
- ▶ npm install
- ▶ npm run dev

▶ **Dependencies:**

- ▶ [Nodejs 5.0+](#)
- ▶ [Truffle](#)
- ▶ [Ganache](#)
- ▶ Le test est validé, il vous reste d'ajouter le smart contract et le fichier de déploiement



## Atelier 5: Construire votre DApp pour localiser vos animaux



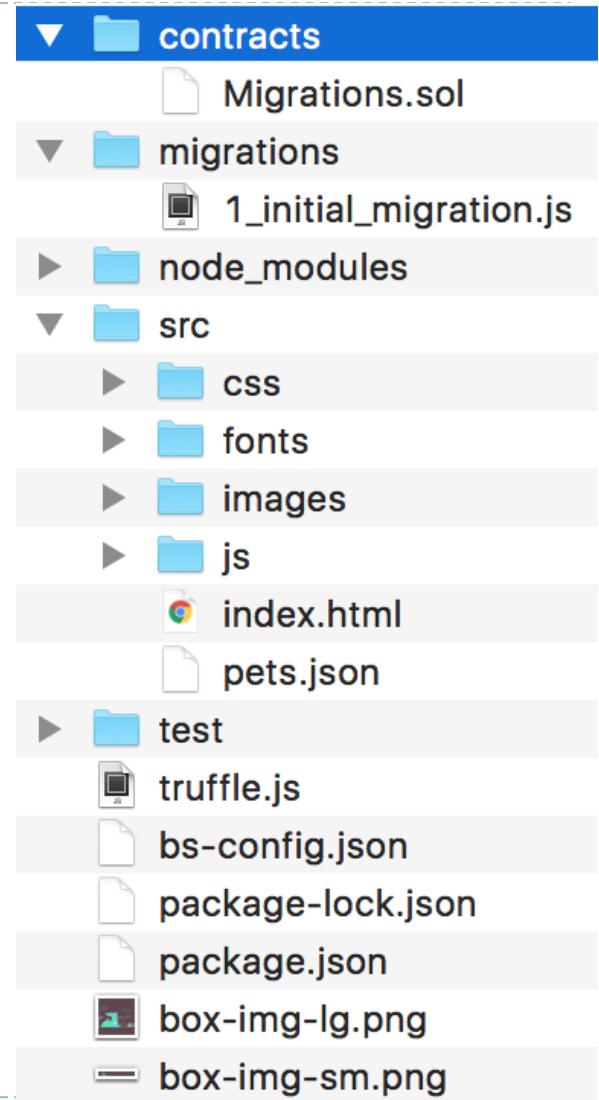
# Installation des dépendances

---

- ▶ NodeJS
- ▶ Node Package Manager (NPM)
- ▶ Truffle Framework
- ▶ Ganache ou Ganache-cli
- ▶ Metamask
- ▶ Un Editeur de texte
  - ▶ Sublime text, Atome, Coda, TextMate, Notepad++
- ▶ ou bien un IDEs
  - ▶ VS Code, Eclipse YAKINDU Solidity Tools, IntelliJ-Solidity, etc.
- ▶ Maintenant, créons un répertoire de projet pour notre dApp dans la ligne de commande comme ceci:
  - ▶ \$ mkdir election
  - ▶ \$ cd election

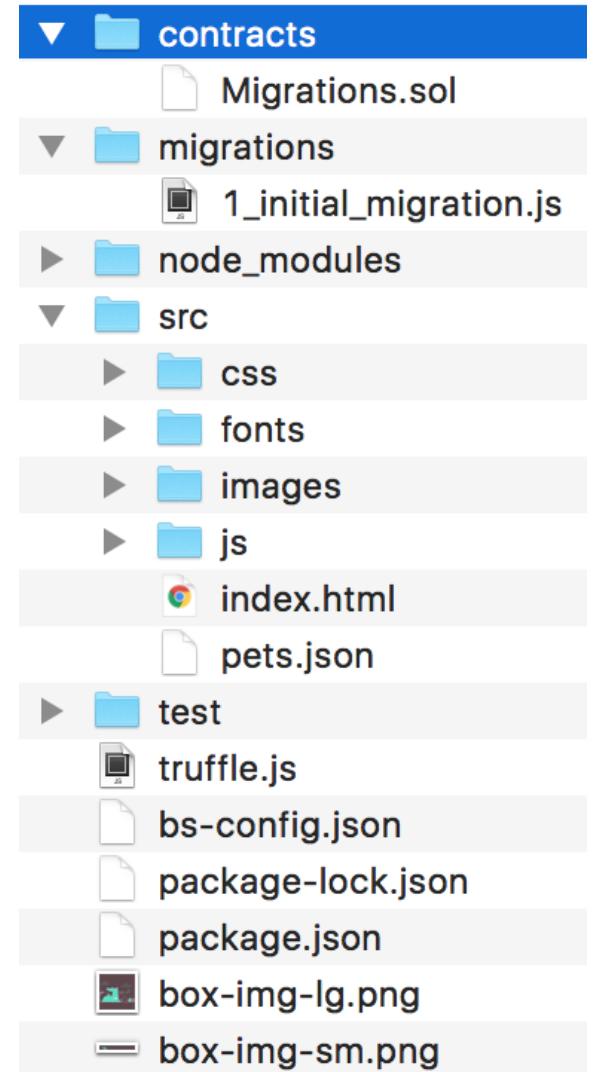
# Installation de la box

- ▶ À partir de votre répertoire de projet « election », installez la boîte Pet Shop à partir de la ligne de commande, comme suit:
  - ▶ \$ truffle unbox pet-shop
- ▶ répertoire « **contracts** » :
  - ▶ ici se trouvent tous les contacts intelligents.
  - ▶ Le fichier « Migration.sol » gère nos migrations vers la blockchain.
- ▶ Le dossier « **migrations** »:
  - ▶ ici résident tous les fichiers de migration.
  - ▶ Chaque fois que nous déployons des smart contrats sur la blockchain, nous mettons à jour l'état de la blockchain et nous avons donc besoin d'une migration.



# Les dossiers du projet

- ▶ **répertoire « node\_modules »:**
  - ▶ c'est le répertoire de toutes nos dépendances de nœuds.
- ▶ **répertoire « src »:**
  - ▶ Ici nous allons développer notre application côté client.
- ▶ **répertoire de « test »:**
  - ▶ Ici, nous allons écrire nos tests pour nos contrats intelligents.
- ▶ **fichier « truffle.js »:**
  - ▶ il s'agit du fichier de configuration principal de notre projet Truffle



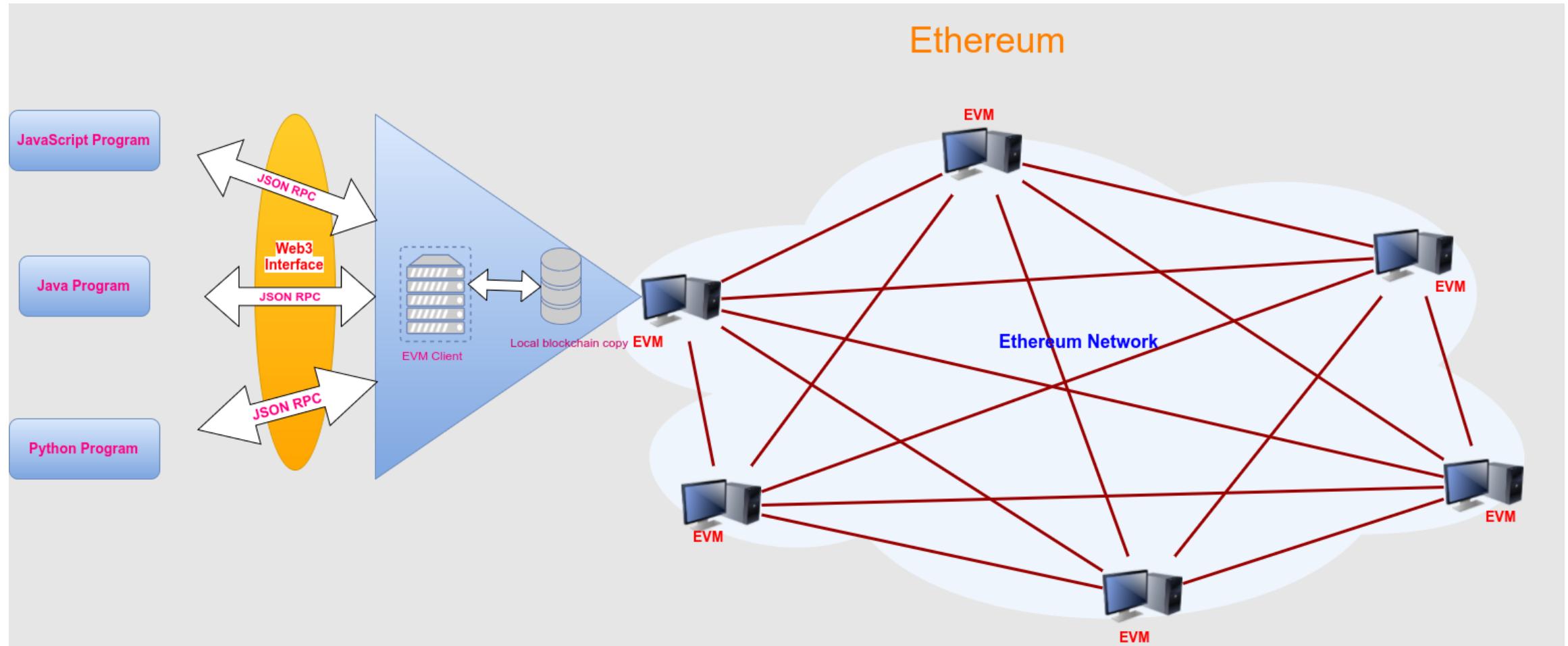




## Atelier 5: Développer le contrat Ethereum en utilisant Python Flask

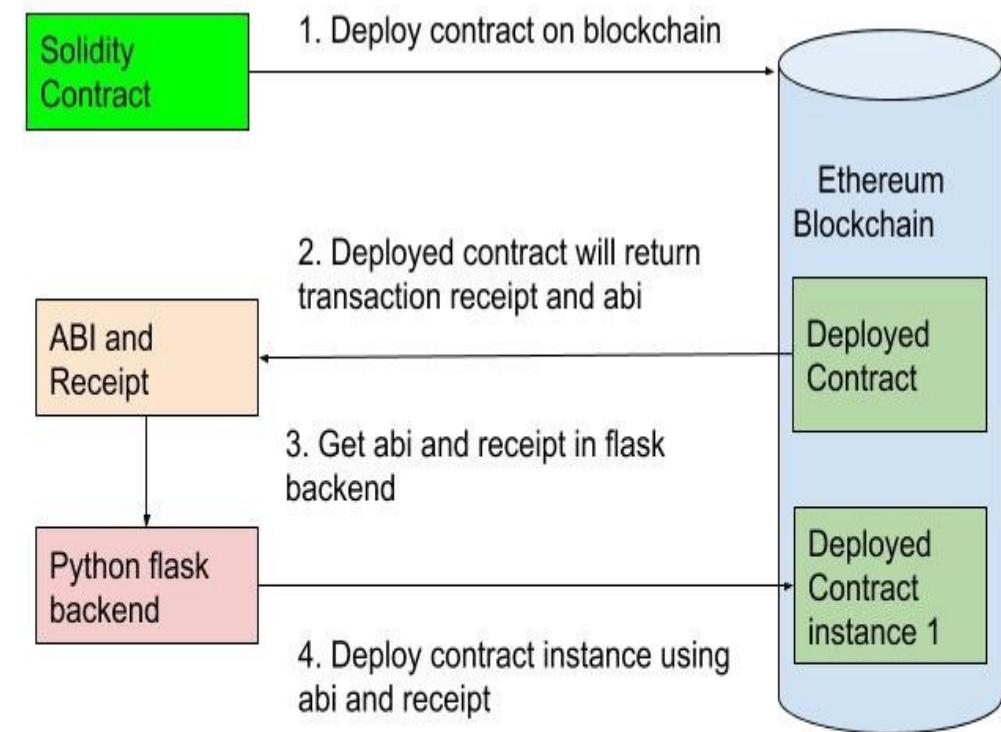


# Introduction à web3.py



# Smart Contrat en utilisant Python

- ▶ Nous allons écrire un smart contrat pour la persistance des données utilisateur sur la blockchain.
- ▶ Nous allons utiliser python web3 pour créer et déployer un contrat intelligent.
- ▶ Nous allons interagir avec smart contrat en utilisant l'API Flask pour stocker certaines données/informations.



# Etape 1: installation de l'environnement

---

- ▶ Installer python pour votre OS
  - ▶ <https://www.python.org/download/releases/3.0/>
- ▶ Préparer votre environnement virtuel de python
  - ▶ Virtualenv conserve vos packages Python dans un environnement virtuel localisé dans votre projet, au lieu de vous obliger à installer vos packages à l'échelle du système.
  - ▶ <https://virtualenv.readthedocs.io/en/latest/>
  - ▶ \$ pip install virtualenv
  - ▶ \$ virtualenv -p /usr/bin/python3.6 venv (sous windows : virtualenv env) (vérifier le dossier et les fichiers d'activation « bat » sont créés lancer activate.bat)
  - ▶ \$ source venv/bin/activate
- ▶ Installer ethereum test chain : ganache
  - ▶ \$ npm install -g ganache-cli
- ▶ Installer l'API Web3.py pour Python
  - ▶ pip3 install web3
- ▶ Install solidity compiler pour python
  - ▶ pip3 install py-solc
- ▶ Installer Flask framework pour python
  - ▶ \$ pip3 install flask
- ▶ Installer Flask Restful
  - ▶ \$ pip3 install flask-restful
- ▶ Installer Flask Marshmallow pour la sérialisation-désérialisation des objets
  - ▶ \$ pip3 install flask-marshmallow

# Ecrire Le Smart contract

- ▶ Lancer le serveur Ethereum Test blockchain
  - ▶ \$ ganache-cli
- ▶ Ecrire votre premier « smart contract » en langage solidity
  - ▶ Vous pouvez télécharger directement le smart contract sur les fichiers « user.sol » et « stringUtils.sol » lien suivant:
    - ▶ <https://github.com/hakiri/Tuto-Blockchain>
  - ▶ Lancer la console pour exécuter
    - ▶ set FLASK\_APP=app.py
    - ▶ Flask run

```
C:\Users\hakir>ganache-cli
Ganache CLI v6.4.1 (ganache-core: 2.5.3)

Available Accounts
=====
(0) 0x96f5d7bba53909686c9a59f71c9af35fad51c9ad (~100 ETH)
(1) 0x768e762d25083e8c60e65f3bb4c5e3950e6abbd (~100 ETH)
(2) 0xc7fd38e8a575882035a7862b020603f871f18cf2 (~100 ETH)
(3) 0xcdbe26611de2b9b1e1f910526f0405389dda1022 (~100 ETH)
(4) 0xdf8aa016002df72f0695f5637d26aa659e21c206 (~100 ETH)
(5) 0xdeb270c50ea5bf4c9767863d4de9f42b48c0ede4 (~100 ETH)
(6) 0x69f2f2f8170351c096ac6b6728e145be62c0f01e (~100 ETH)
(7) 0xa8b92fad624ff463a395a99d647e314b316b992 (~100 ETH)
(8) 0x033601e4ed42ef63c346297b87f471ec73312094 (~100 ETH)
(9) 0x655af441789990373180506d0865b97af3465c84 (~100 ETH)

Private Keys
=====
(0) 0x0231e968339dbd06c0009a8b1a83ab4eaf8482b2377a5f791c7e85516cbe90d6
(1) 0x6b6d6fac30caf3e6335e7ebb25da03511b9e5955aa923b0def70d428f4689f0e
(2) 0xe8aed4388b89745ca8af5a17ba2705e6c965d24ed64339b86307746bc7e4de9c
(3) 0xf25fe5f2f8636dc17e336204bf26a6442b05c0f43d27374ec06f53db7ecb8d97
(4) 0x60018134118e2817312749df3e4a0a204dab2270385b458fa597cef0705ded7
(5) 0x848fbec898e18ee8a7722134e85f508c358084afffc9c8938206a693990e5aacc
(6) 0x859757a1df13d498217cd082905c327c53bf815076563d7b2d0d441cb01e3f9a
(7) 0xdb5b0e8bd168f7ad6e4716b17b56326983021e169d7ecc2bb361143a174d296e
(8) 0x347d35b11a8bab699de6eb95392cd5fc0acfe82bdb2e7a504b1651adc539d07b
(9) 0x301e2e00f4e57fb171603789889d1c61425423f807feee7c4baa245d905462a6

HD Wallet
=====
Mnemonic:      whale leopard stand capable surface sight earth access drastic join swarm sport
Base HD Path:  m/44'/60'/0'/0/{account_index}

Gas Price
=====
20000000000

Gas Limit
=====
6721975

Listening on 127.0.0.1:8545
```

# Execution

```
C:\Users\hakir\Desktop\Tuto Blockchain\Blockchain Python\basic contract>set FLASK_APP=main  
  
C:\Users\hakir\Desktop\Tuto Blockchain\Blockchain Python\basic contract>flask run  
* Serving Flask app "main"  
* Environment: production  
WARNING: Do not use the development server in a production environment.  
Use a production WSGI server instead.  
* Debug mode: off  
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)  
127.0.0.1 - - [15/Apr/2019 22:30:16] "GET / HTTP/1.1" 404 -  
127.0.0.1 - - [15/Apr/2019 22:30:18] "GET /favicon.ico HTTP/1.1" 404 -  
127.0.0.1 - - [15/Apr/2019 22:31:13] "POST /blockchain/user HTTP/1.1" 400 -  
127.0.0.1 - - [15/Apr/2019 22:31:47] "GET / HTTP/1.1" 404 -  
127.0.0.1 - - [15/Apr/2019 22:31:49] "GET / HTTP/1.1" 404 -  
127.0.0.1 - - [15/Apr/2019 22:39:48] "GET / HTTP/1.1" 404 -
```



## Method Not Allowed

The method is not allowed for the requested URL.

## Atelier 6 : Développer une application IoT sécurisée dans la blockchain

# Introduction

---

- ▶ **Imaginez une machine qui**
  - ▶ contacte de manière autonome les fournisseurs de services et
  - ▶ pose une demande de service en cas de problème, effectue le self-service et assure la maintenance;
- ▶ **Grâce à l'Internet des objets,**
  - ▶ Plusieurs entreprises s'emploient déjà activement à transformer le milliard d'appareils IoT connectés à Internet, ce qui n'était pas adapté aux ordinateurs.
  - ▶ Plusieurs cas d'utilisation sont possible, en particulier lorsque la puissance de l'IoT est combinée à celle d'autres technologies, telles que le cloud computing, le big data et l'apprentissage automatique.
- ▶ **Les objets IoT nécessitent le renforcement de la sécurité de leurs interactions sans confiance entre eux et les autres fournisseurs de services sur Internet.**
- ▶ **la technologie blockchain permet de nouveaux types d'interactions sans confiance.**

# Introduction

---

- ▶ Dans cet atelier, nous allons utiliser une API Web3 pour javascript
- ▶ La documentation officielle : <https://web3js.readthedocs.io/en/1.0/>
- ▶ Il y a aussi API Web3 pour python
- ▶ La documentation officielle: <https://web3py.readthedocs.io/en/stable/>

# Environnement

---

- ▶ Nous utilisons le Webpack de truffle
  - ▶ Truffle unbox webpack
  - ▶ Le webpack arrive avec deux smart contracts : machine-maintenance-eth → contracts → Metacoin.sol, ConvertLib.sol
  - ▶ Nous n'avons pas besoin de ces contracts, vous allez les supprimer
  - ▶ Copier le fichier maintenance.sol dans le dossier (github)
  - ▶ Configurer le fichier truffle-config.js pour choisir votre réseau de déploiement
  - ▶ Changer index.js avec l'emplacement du nouveau fichier maintenance.json
    - ▶ import metaCoinArtifact from "../../build/contracts/Maintenance.json";
- ▶ Le code source de cette application se trouve dans github

# Compilation

---

- ▶ **Compilation, migration et déploiement**
  - ▶ Commands:
    - ▶ Compile:      `truffle compile`
    - ▶ Migrate:      `truffle migrate`
    - ▶ Test contracts:      `truffle test`
    - ▶ Run dev server:      `cd app && npm run dev`
    - ▶ Build for production: `cd app && npm run build`
- ▶ Assurez-vous d'avoir un client Ethereum comme GANACHE sur `http://localhost:8545`.
- ▶ Naviguer vers : `localhost:8081/index.html`

# Préparer votre environnement

---

- ▶ **Etape 1 : installer ganache-cli et ganache GUI**
  - ▶ npm install -g ganache-cli
  - ▶ Vérifier que c'est bien installé en tapant dans un terminal: c:\> ganach-cli
  - ▶ Vous pouvez aussi utiliser Ganache GUI
- ▶ **Etape 2: installer truffle-contract**
  - ▶ npm install truffle-contract
  - ▶ npm install -g truffle-expect truffle-config web3 truffle-default-builder
- ▶ **Etape 3: telecharger web3.min.js & truffle-contract.min.js.**
  - ▶ <https://github.com/trufflesuite/truffle-contract/tree/master/dist>
- ▶ **Etape 4: telecharger le code source de cette application atelier 04**
  - ▶ <https://github.com/hakiri/Tuto-Blockchain.git>
- ▶ **Etape 5: utiliser VS Code pour naviguer dans le code**
- ▶ **Déployer le smart contrat et toute l'application**
  - ▶ truffle migrate --network development

# Déploiement de l'application

## Machine Maintenance Dapp

Contract Address	0x6AacB10337ac095d52EF0133bE413EEB115Ad3A2
Machine Registered	0
Service Requests	0

view recent service requests. Click on a machine-ID to view machine details.

### Register New Machine

Machine-ID	<input type="text"/>
Machine Name	<input type="text"/>
Purchase Date	1556218746430
Manufacturer	<input type="text"/>

**Register**

Machine-ID

Machine Name

Purchase Date

Owner

Manufacturer

Request for new service.

Machine-ID

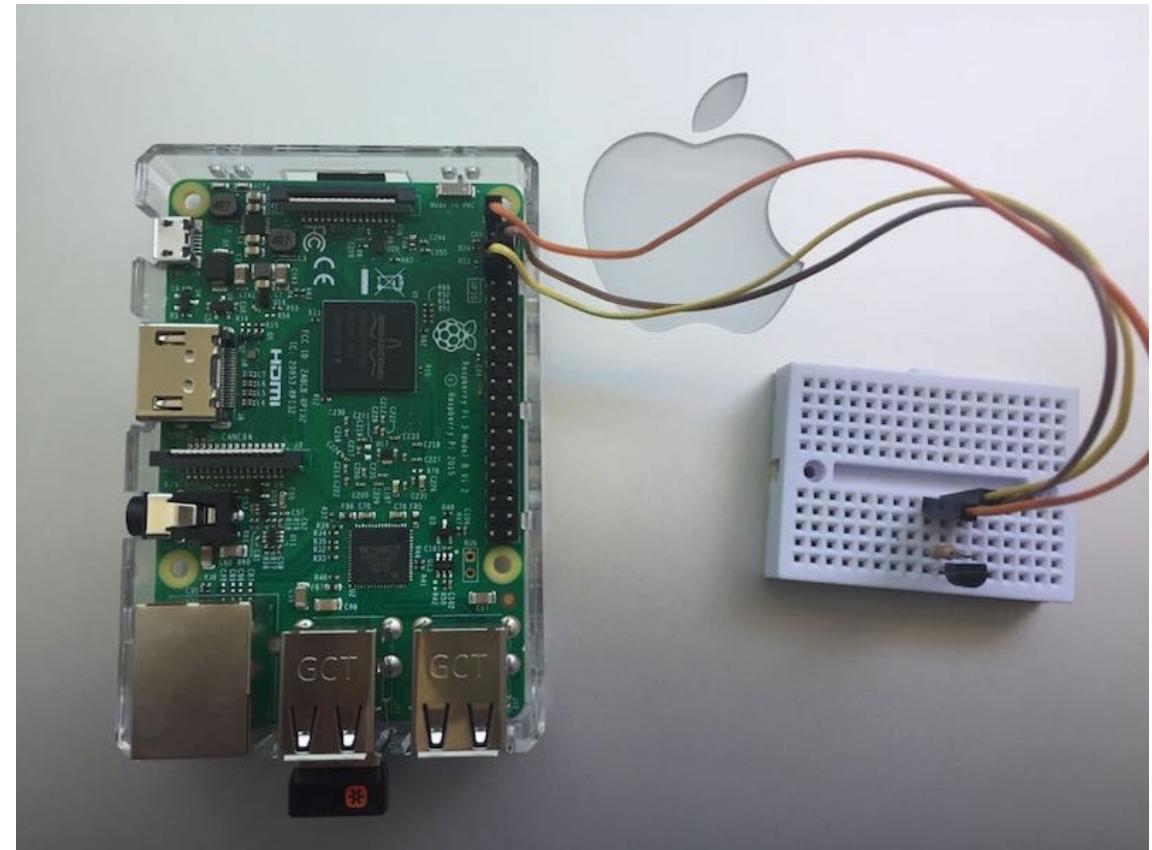
Time Stamp 1556218746430

Remarks

**Request Service**

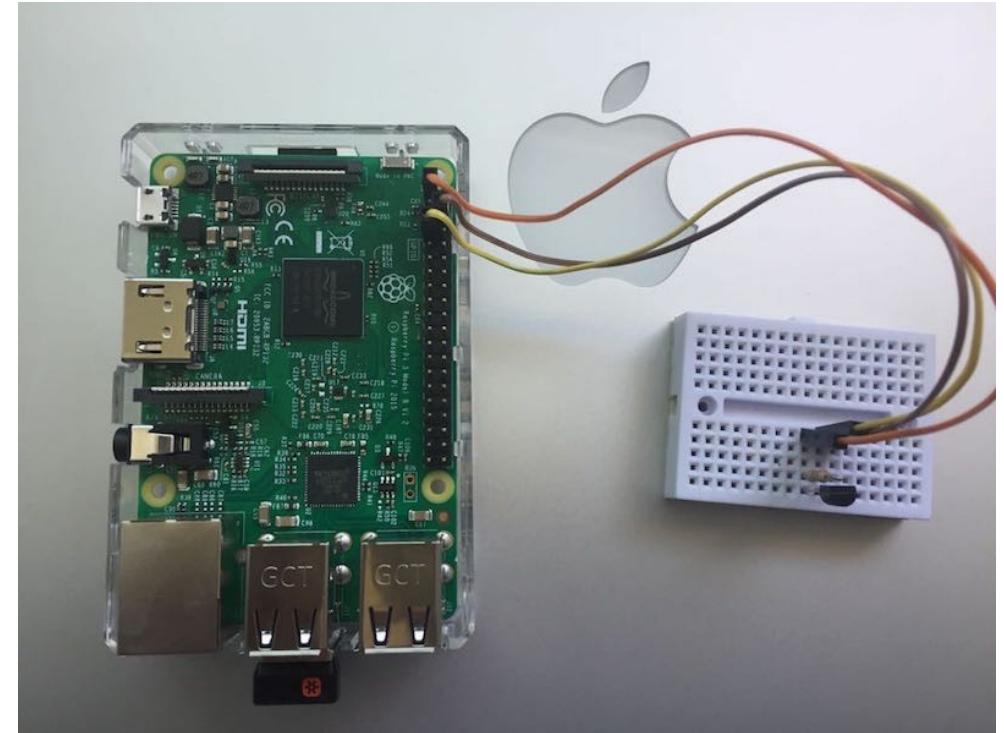
# Cas utilisation : Machine connectée

- ▶ Implémenter un service de contrôleur s'exécute sur le périphérique IoT Raspberry Pi.
- ▶ Le service de contrôle surveille la température et les niveaux de vibration des différentes parties de la machine.



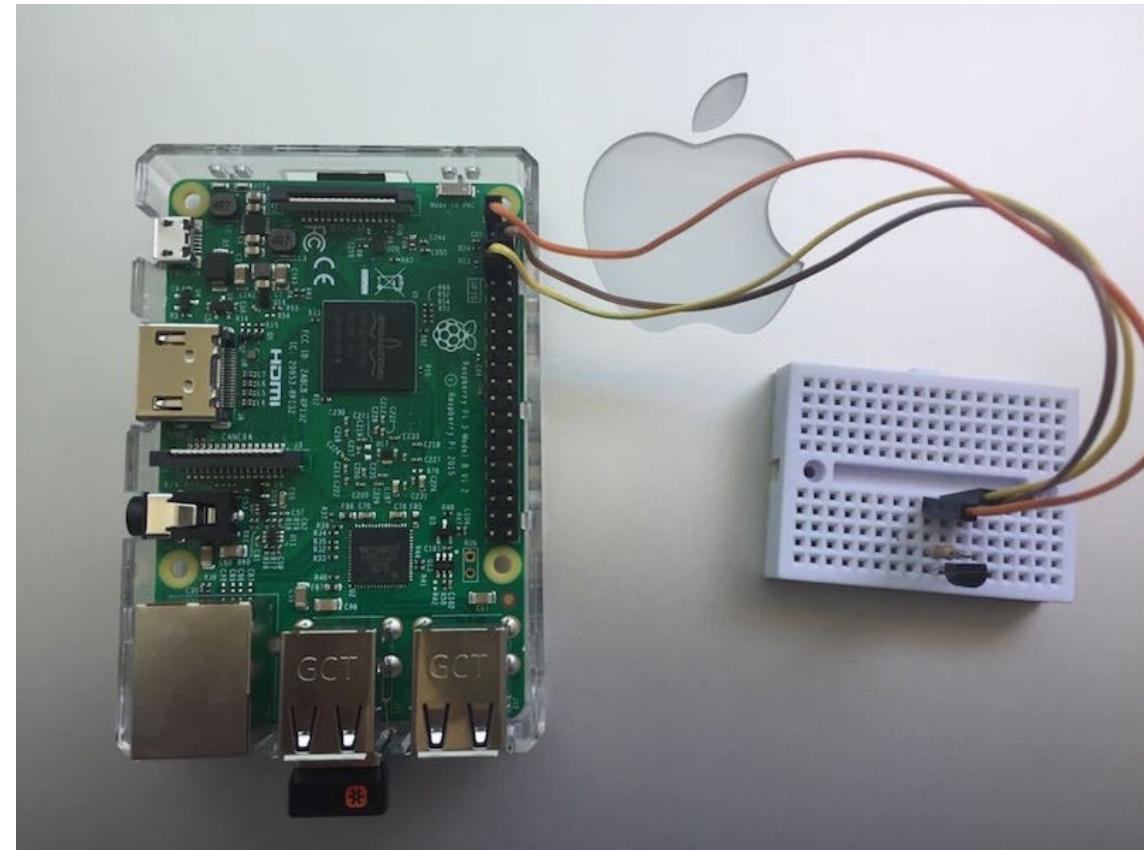
# Cas utilisation : Machine connectée

- ▶ Différentes règles sont définies pour déterminer si une demande de service de machine ou une commande de remplacement de pièce doit être passée.
- ▶ Par exemple, si les niveaux de vibration de la machine dépassent un certain nombre de fois un seuil prédéfini, le service du contrôleur envoie une demande de service de la machine.



# Cas utilisation : Machine connectée

- ▶ De même, si les niveaux de température d'une partie de la machine dépassent un certain nombre de fois un seuil prédéfini, une commande de remplacement de pièce est passée.
- ▶ Pour passer une commande, la machine envoie une transaction à la fonction Commande de pièce du contrat intelligent Remplacement de pièce entre la machine et le fournisseur de pièces en payant le coût de la pièce en devise cryptographique, c'est-à-dire les éthers.



# Préparer raspberry pi

---

## ▶ NPM packages

- ▶ \$ sudo apt update
- ▶ \$ sudo apt install nodejs npm
- ▶ \$ nodejs --version
- ▶ \$ npm --version

## ▶ Python packages

- ▶ \$ sudo apt install python-minimal
- ▶ \$ sudo apt install gcc
- ▶ \$ sudo apt install virtualenv # optional but recommended
- ▶ \$ sudo apt install libpython-dev
- ▶ \$ sudo apt install libssl-dev
- ▶ \$ pip install ethjsonrpc

# Python Script pour le capteur de température

```
1 from ethjsonrpc import EthJsonRpc
2 import time
3 import sys,traceback
4 import datetime
5 import glob
6
7
8 base_dir = '/sys/bus/w1/devices/'
9 device_folder = glob.glob(base_dir + '28*')[0]
10 device_file = device_folder + '/w1_slave'
11
12
13 contract_address = '0x016D9579a34F7855D5A954d127469260e7F3A5a4'
14 machineID = 123
15
16 # Connect to Blockchain network -- Ganache
17 #
18 c = EthJsonRpc('192.168.1.4', 8101)
19 tempCount = 0
20 tempThreshold = 20 #Celcius
21 tempCountThreshold = 10
22 while True:
23     f = open(device_file, 'r')
24     lines = f.readlines()
25
```

# Python Script pour le capteur de température

```
26 ▼    while lines[0].strip()[-3:] != 'YES':
27        time.sleep(0.2)
28        lines = read_temp_raw()
29 equals_pos = lines[1].find('t=')
30 ▼ if equals_pos != -1:
31        temp_string = lines[1][equals_pos+2:]
32        temperature = float(temp_string)/1000.0
33 print "Temp:", temperature, "C"
34 if(temperature>tempThreshold):
35        tempCount = tempCount + 1
36 ts = int(time.time())
37        print ("ts: ", ts)
38 ▼ if(tempCount>tempCountThreshold):
39        c.call_with_transaction(c.eth_coinbase(), contract_address,
40                                '|requestService(uint256,uint256,string)', [ts, machineID, 'High'])
41        tempCount = 0
42        time.sleep(1):
43
```



Merci pour votre attention