Introduction à la

Blockchain Ethereum

et aux

Smart Contracts

Jean-Patrick GELAS
Café des développeurs
Université Claude Bernard – Lyon 1
Mars 2019

jp@cafedev:~$ whoami

```
{
    "first_name" : "Jean-Patrick",
    "last_name" : "Gelas",
    "job" : "Maître de conférences",
    "locations" : [ "Université Claude Bernard - Lyon 1", "Avalon/LIP/INRIA/ENS Lyon" ],
    "url" : "https://perso.univ-lyon1.fr/jean-patrick.gelas",
    "email" : "jean-patrick.gelas@univ-lyon1.fr",
    "github" : "https://github.com/jpgelas",
    "hobbies" : [ "skydive", "wingsuit" ]
}
```

Agenda

Nous ne parlerons pas de ...

- Comment devenir **crypto millionaire** ! :-)
- Comment *miner* de la crypto monnaie
- Plateformes d'échanges (*Binance, Kraken,...*)
- Algorithmes de consensus (*PoW, PoS, PoA, DPoS,...*)

Agenda

- Rappel sur le fonctionnement d'une blockchain,
- La blockchain Ethereum
- Les Smart Contracts
- Les outils pour déployer et interagir un Smart Contract,
- Introduction aux *Dapps*.

Blockchain : Rappel

- Structure de données simple
- La technologie Blockchain : « Base de données » sécurisées et décentralisées.

Démo : https://anders.com/blockchain/

Les mineurs

- Héberge une copie de la blockchain
- Ajoutent de nouvelles liste de transactions (*i.e.* des blocs) à la chaîne.
- Vérifient l'intégrité de la blockchain
- Génèrent de nouveaux *coins*
- (Exécutent les Smart Contracts)

Ethereum : Définition

« Protocole d'échanges décentralisés permettant la création par les utilisateurs de contrats intelligents grâce à un langage Turing-complet. »

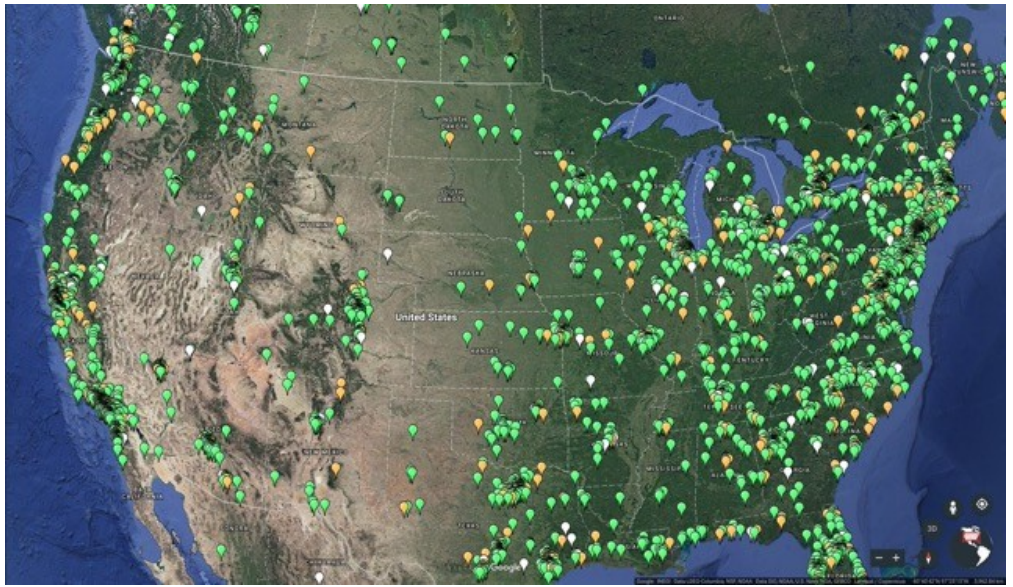
- Wikipedia

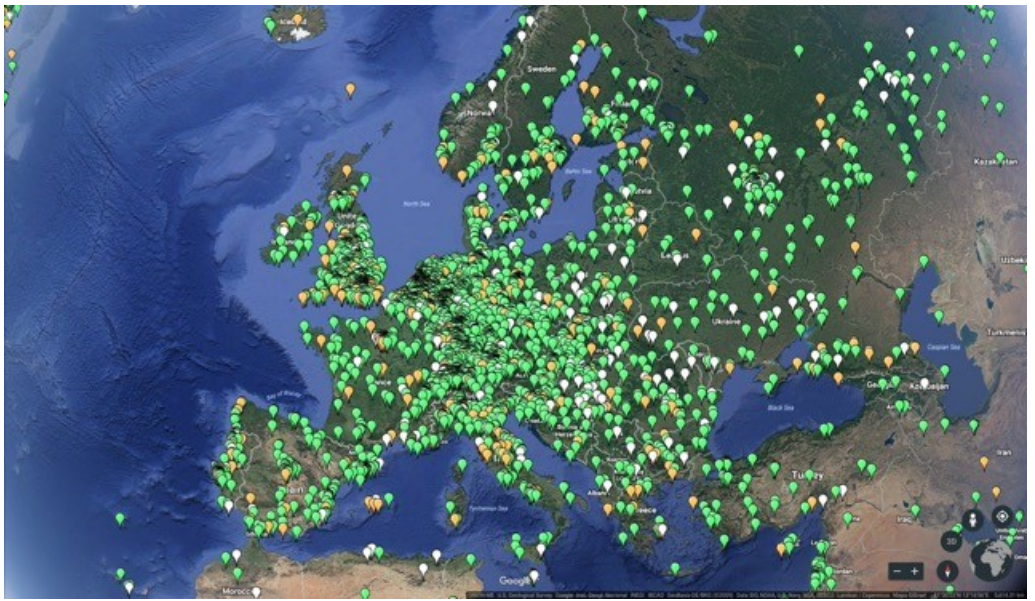
Ethereum

- Blockchain de seconde génération.
- Développée par *Vitalik Buterin*, lancée en juillet 2015.
- Fréquence moyenne des blocs : 14-15 secondes
- Taille des blocs : dynamique
- Symbole boursier : ETH
- Quantité maximale : non limitée

Infrastructure Ethereum

9000 nœuds contre 7000 pour Bitcoin (mars 2019).



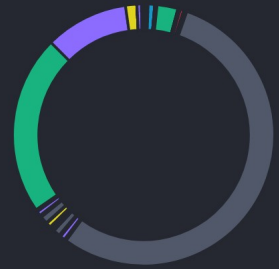


Source: https://twitter.com/peter_szilagyi/status/887272506914213888 - 18/07/2017

https://www.ethernodes.org

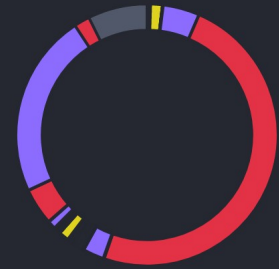
Network number 1 Last updated a few seconds ago

CLIENTS



Clients

Client Versions

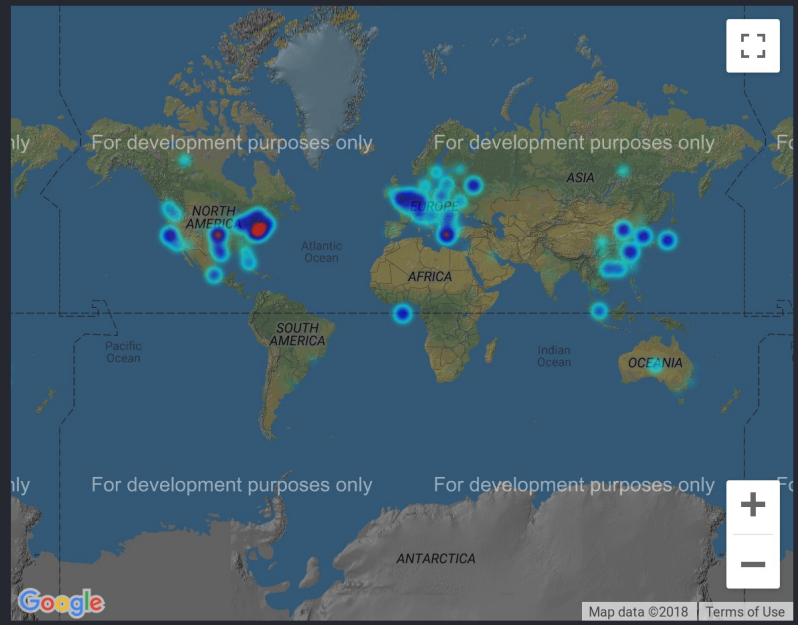


OS

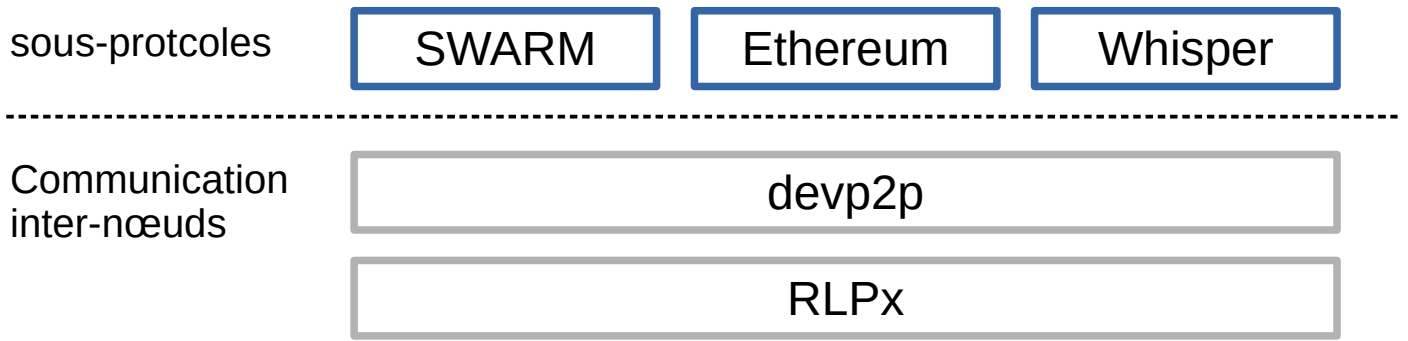
Like what you see? Support the node explorer!

COUNTRIES

Total	13320 (100%)
United States	5738 (43.08%)
China	1639 (12.30%)
Canada	1041 (7.82%)
Germany	587 (4.41%)
Russian Federation	476 (3.57%)
United Kingdom	419 (3.15%)
Netherlands	288 (2.16%)
Korea, Republic of	246 (1.85%)
France	236 (1.77%)
Greece	215 (1.61%)



Architecture Ethereum



- **Swarm :** Un système de fichiers décentralisé proche de IPFS permettant de servir les Dapps.
- **Ethereum** : La blockchain que nous connaissons permettant de stocker la logique des *Dapps* (smart-contract).
- **Whisper** : Un protocole de communication décentralisé pour communiquer entre les *Dapps*.
- ***devp2p***: Abstraction de la couche matériel pour communiquer entre les nœuds.

Smart Contract

- Programme autonome
- Déployé et répliqué
- Non modifiable
- Adapté pour gérer des transactions

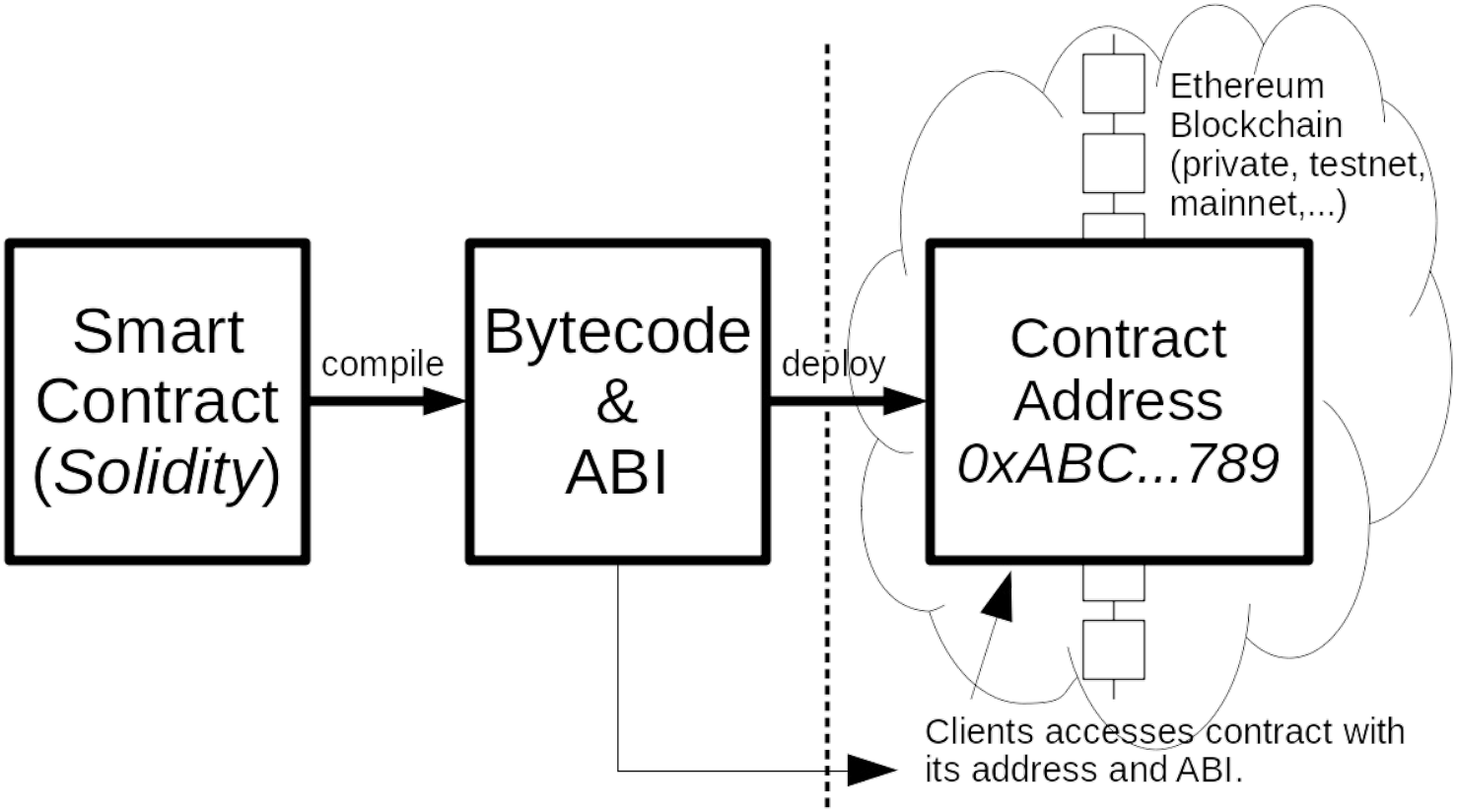
Smart Contract

- Du *bytecode* stocké dans la blockchain
- Rédigé dans un langage de haut niveau : *Solidity*
- Compilé (solc)
- Accessible via une adresse codée sur 160 bits
  Exemple : `0xabc7656ec7ab88b098defb751b7401b5f6d89789`
- Exécuté dans l'Ethereum Virtual Machine (EVM)

Smart Contract

- Le *bytecode* est déployé sur Ethereum dans une transaction (dans le champ *data* de la transaction).
- Une fois sur la blockchain le contrat ne peut être ni modifié ni supprimé.
- Il est accessible sur tous les nœuds à jour.
- Pour interagir avec le contrat il faut une machine virtuelle qui interprète le *bytecode*.
- L'EVM (Ethereum Virtual Machine) mise à disposition par les clients Ethereum qui gère un nœuds (ex : Geth, Parity, Cpp-Ethereum,...)

Cycle de vie du Smart Contract



Code machine

"opcodes": "PUSH1 0x80 PUSH1 0x40 MSTORE
CALLVALUE DUP1 ISZERO PUSH2 0x10 JUMPI PUSH1 0x0
DUP1 REVERT JUMPDEST POP PUSH1 0x40 MLOAD PUSH1
0x20 DUP1 PUSH2 0x487 DUP4 CODECOPY DUP2 ADD
PUSH1 0x40 SWAP1 DUP2 MSTORE SWAP1 MLOAD PUSH1..."

"object":
"6080604052348015610010576000 80fd5b5060405160
208061048783398101604090815290516000805460016
0a060020a031916331780825560160a060020a031681
526001602081905292902091909155 60ff81166100606..."

Clients Ethereum

- Coté client on a un *provider* qui se connecte à un nœud en particulier et qui communique avec l'EVM via des requêtes au format JSON-RPC.
- Les clients Ethereum proposent une API avec un ensemble de méthodes JSON-RPC.

https://github.com/ethereum/wiki/wiki/JSON-RPC#json-rpc-methods

JSON-RPC

curl -X POST --data '{"jsonrpc":"2.0","method":"**eth_blockNumber**","params":[],"id":1}' https://mainnet.infura.io/v3/*64da4bf9a73a...a23ec751b9ef37c*

{"jsonrpc":"2.0","id":1,"result":"**0x7132b0**"}

0x7132b0 => 7,418,544 (22/03/2019 13h35 (Paris))

curl -X POST --data '{"jsonrpc":"2.0","method":"**web3_clientVersion**","params":[],"id":67}' http://127.0.0.1:8545

{"id":67,"jsonrpc":"2.0","result":"**EthereumJS TestRPC/v2.3.3/ethereum-js**"}

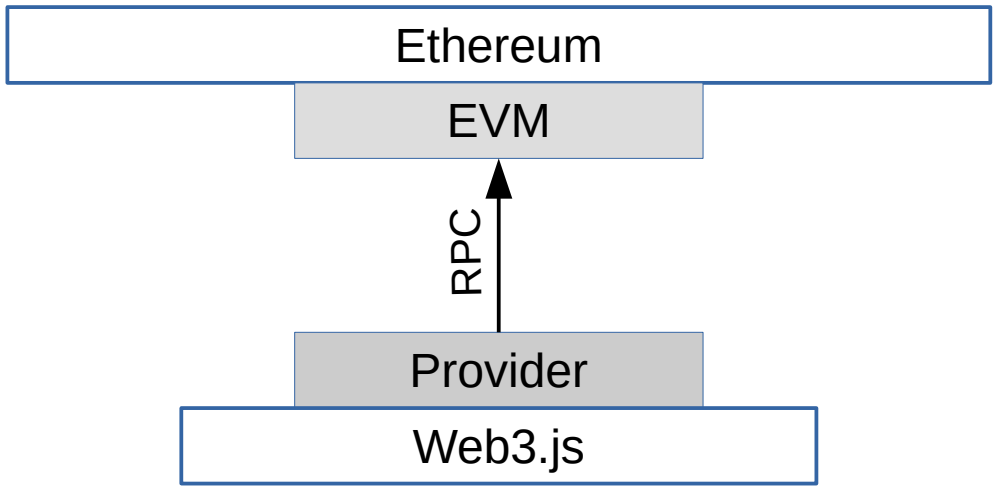
Communiquer avec Ethereum

- Envoyez des requêtes JSON-RPC avec son propre provider n’est pas forcément simple à mettre en place et à manipuler.
- Un des projets les plus utilisés sur Ethereum propose d’englober ce provider et ces requêtes avec une API JavaScript (existe aussi dans d’autres langages). Il s’agit de *Web3*.
- Une alternative populaire à Web3 est *ethers.js*

Web3.js / Ethers.js

- Bibliothèques JavaScript simples d’utilisation permettant d’interagir avec un nœud.
- La première chose à faire d’ailleurs lorsque l’on utilise Web3 est de lui donner un *provider* avec l’adresse du nœud à contacter.
- Idéalement le nœud auquel se connecte Web3 tourne sur la machine de l’utilisateur.
- Mais de plus en plus les utilisateurs se connectent à des nœuds distants grâce à des services comme Infura (https://infura.io/)

Web3 pour interagir avec la blockchain

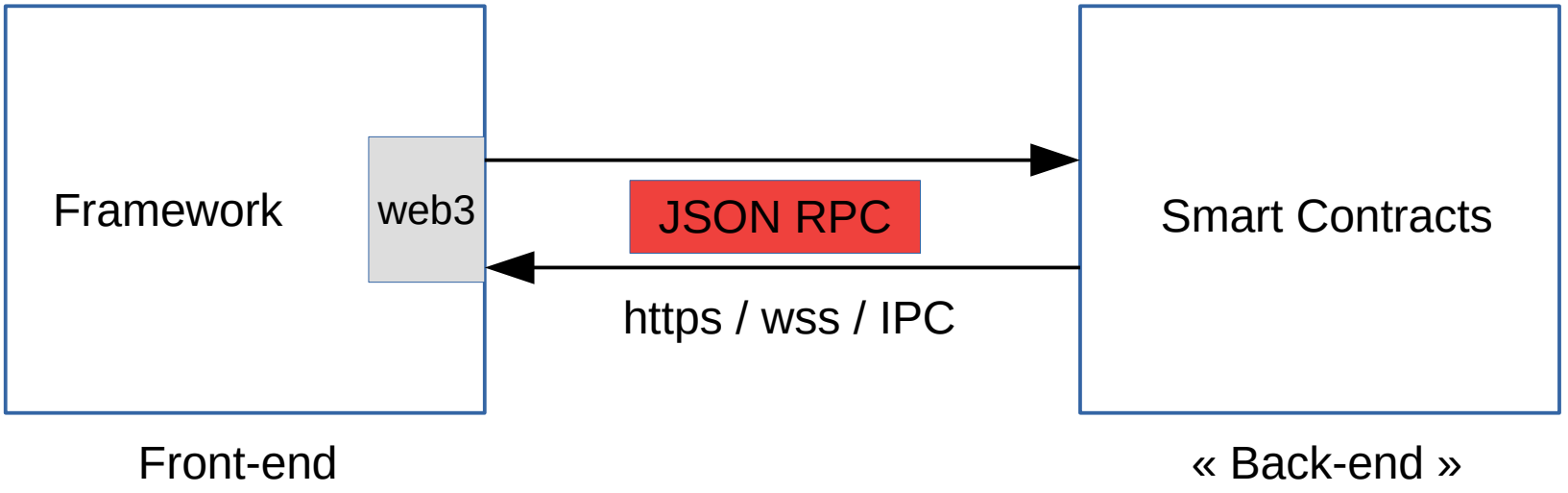


JavaScript API
(Client/Serveur)

Dapps

- Application accessible de façon décentralisée (en principe).
  - Pas de serveur centralisé
  - ni pour gérer une « base de données »,
  - ni pour servir l’application,
  - ni pour communiquer entre plusieurs applications.
- Le projet Ethereum propose une solution pour chacun de ces points
- *Swarm*, *Whisper* ne sont pas encore suffisamment mature...
- Une Dapp se “limite” donc à la connexion entre une application communiquant avec des smart-contracts sans s’appuyer sur un serveur pour gérer une base de données centralisée.

Dapp


Framework
web3
JSON RPC
https / wss / IPC
Smart Contracts
Front-end
« Back-end »

Le langage Solidity

- Langage de haut niveau
- Influencé par C++, Python et Javascript.
- Typé statiquement.
- Supporte l'héritage,
- l'appel à des bibliothèques,
- la définition de type complexe par les utilisateurs.

Hello World

```
pragma solidity ^0.4.18;

contract Hello {
    string message = "Default message";

    function getMessage () public view returns (string) {
        return message;
    }
    function setMessage (string _message) public payable {
        message = _message;
    }
}
```

Simple Token

```
pragma solidity ^0.4.20;

contract MyToken {
    /* This creates an array with all balances */
    mapping (address => uint256) public balanceOf;

    /* Initializes contract with initial supply tokens to the creator of the contract */
    function MyToken(
        uint256 initialSupply
        ) public {
        balanceOf[msg.sender] = initialSupply;              // Give the creator all initial tokens
    }

    /* Send coins */
    function transfer(address _to, uint256 _value) public returns (bool success) {
        require(balanceOf[msg.sender] >= _value);           // Check if the sender has enough
        require(balanceOf[_to] + _value >= balanceOf[_to]); // Check for overflows
        balanceOf[msg.sender] -= _value;                    // Subtract from the sender
        balanceOf[_to] += _value;                           // Add the same to the recipient
        return true;
    }
}
```

Nexium (NXC) – Token ERC-20

```
contract Nexium {
    ...
    function Nexium() {
        initialSupply = 100000000000;
        balanceOf[msg.sender] = initialSupply;
        name = 'Nexium';
        symbol = 'NxC';
        decimals = 3;
        burnAddress = 0x1b320000000000000000000000000000000000
    }
    function totalSupply() returns(uint){
        return initialSupply - balanceOf[burnAddress];
    }
    function transfer(address _to, uint256 _value) ...
    function transferFrom(address _from, ...
```

0x (ZRX) – Token ERC-20

```
contract ZRXToken is UnlimitedAllowanceToken {

    uint8 constant public decimals = 18;
    uint public totalSupply = 10**27; // 1 billion tokens
    string constant public name = "0x Protocol Token";
    string constant public symbol = "ZRX";
    ...

    function ZRXToken() {
        balances[msg.sender] = totalSupply;
        ...
```

Ethereum Virtual Machine

- Machine (*quasi*-) Turing complète
- Environnement d'exécution des Smart Contracts
- Émule une machine 256 bits avec des pseudo-registres
- Registres émulés par une *stack* virtuelle.

Ethereum et unités de mesure

- **Ether** (ETH) : le nom de la crypto monnaie
- **Wei** : une fraction d'Ether (1 ETH = 10^{18} Wei)
- **GAS** : unité de mesure en terme de quantité de calcul
- **GAS price** : défini le prix (en GWei) que vous êtes prêt à payer au mineur.
- **GAS limit** : Quantité maximum de GAS que vous êtes prêt à payer pour une transaction.

Coût des instructions

Value	Mnemonic	Gas Used	Subset	Removed from stack	Added to stack	Notes
0x00	STOP	0	zero	0	0	Halts execution.
0x01	ADD	3	verylow	2	1	Addition operation
0x02	MUL	5	low	2	1	Multiplication operation.
0x03	SUB	3	verylow	2	1	Subtraction operation.
0x04	DIV	5	low	2	1	Integer division operation.
0x51	MLOAD	3	verylow	1	1	Load word from memory.
0x52	MSTORE	3	verylow	2	0	Save word to memory
0x53	MSTORE8	3	verylow	2	0	Save byte to memory.
0x54	SLOAD	200		1	1	Load word from storage
0x55	SSTORE	((value != 0) && (storage_location == 0)) ? 20000 : 5000		1	1	Save word to storage.

Analogie

- GAS limit : capacité du réservoir d'une voiture en litre.
- GAS price le prix du litre de carburant.
  - Voiture : 1,50 EUR (prix) par litre (unité)
  - Ethereum : 20 GWei (prix) par GAS (unité)
- Pour remplir le réservoir il faut :
  - 50 litres à 1,50 EUR = 75 EUR
  - 21000 unités de GAS à 20 GWei = 0.00042 ETH

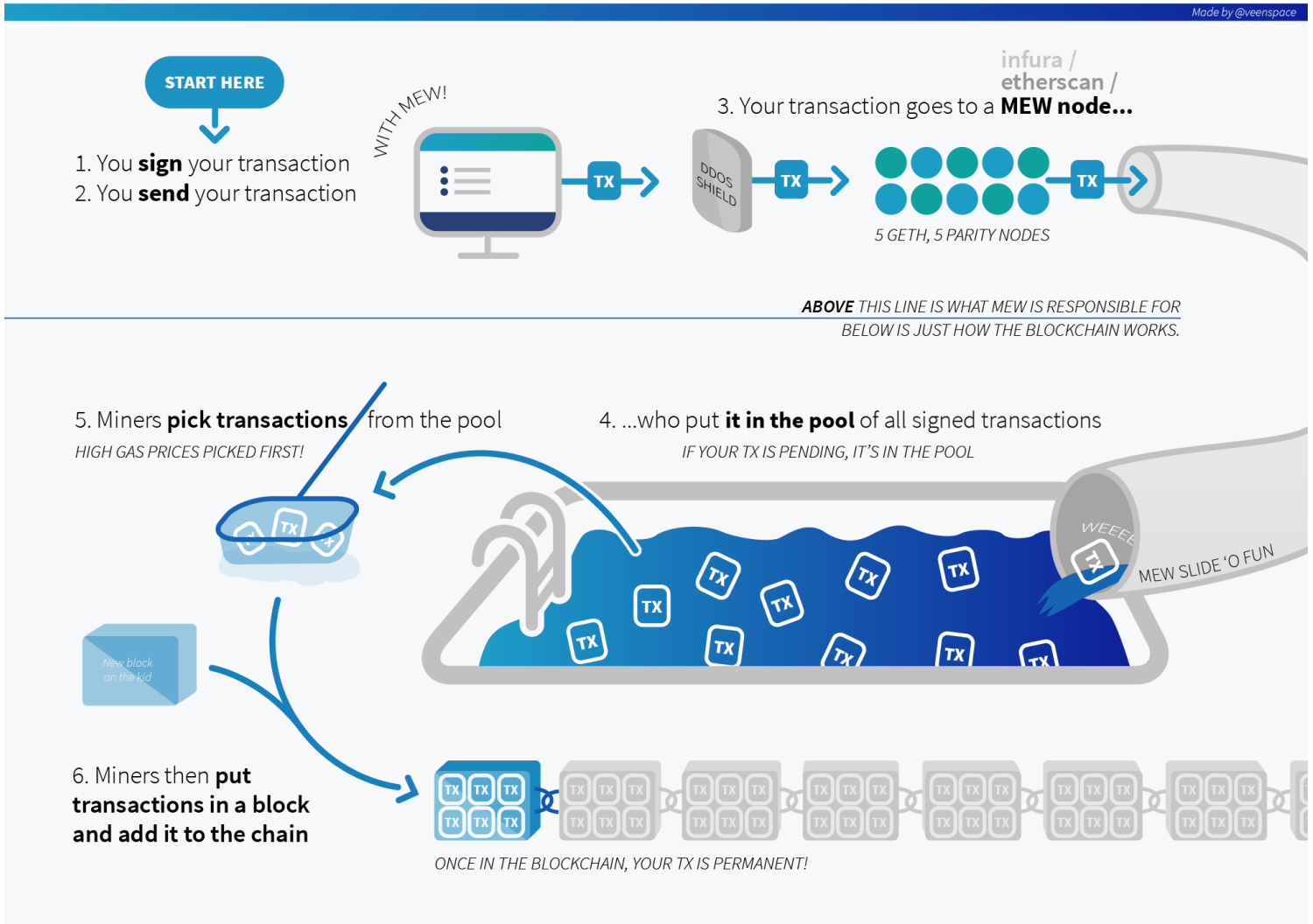
Remarques

- Fixer une *GAS limit* évite de dépenser une fortune en cas de problème.
- La quantité de GAS requise est définie par le coût et la quantité d'instructions exécutées.
- Fixer un *GAS limit* trop petit a peu d'intérêt.

Coût d’une transaction

- Coût total d'une transaction =
  GAS_price x GAS_used
- Priorité aux transactions avec un *GAS_price* élevé.
- Plus l'utilisateur est prêt à payer, plus vite la transaction sera traitée.

MyEtherWallet Behind-The-Scenes



GENERAL

Main Page

Tx Calculator

TxPool Vision

Low Gas Price Watch List

Gas Burners

FAQ

External Links

Std Cost for Transfer	Gas Price Std (Gwei)	SafeLow Cost for Transfer	Gas Price SafeLow (Gwei)	Median Wait (s)	Median Wait (blocks)
$0.015	3.5	$0.011	2.6	30	2

Gas-Time-Price Estimator: For transactions sent at block: 6599460

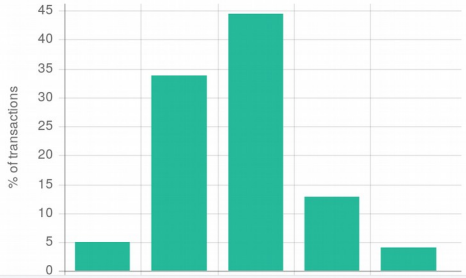
Adjust confirmation time

Avg Time (min)	24.74	Gas Used*	21000
95% Time (min)	61.85	Avg Time (blocks)	98.210721006
Gas Price (Gwei)*	3.5	95% Time (blocks)	245.526802515
Tx Fee (Fiat)	$0.015	Tx Fee (ETH)	0.00007

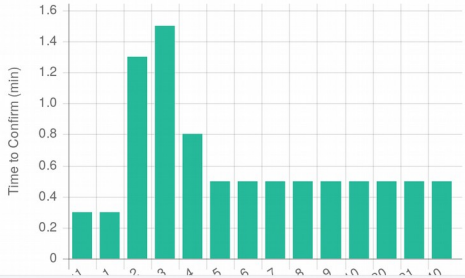
Real Time Gas Use: % Block Limit (last 10)

Last Block: 6599460

Transaction Count by Gas Price



Confirmation Time by Gas Price



Recommended Gas Prices

(based on current network conditions)

Speed	Gas Price (gwei)
SafeLow (<30m)	2.6
Standard (<5m)	3.5
Fast (<2m)	16

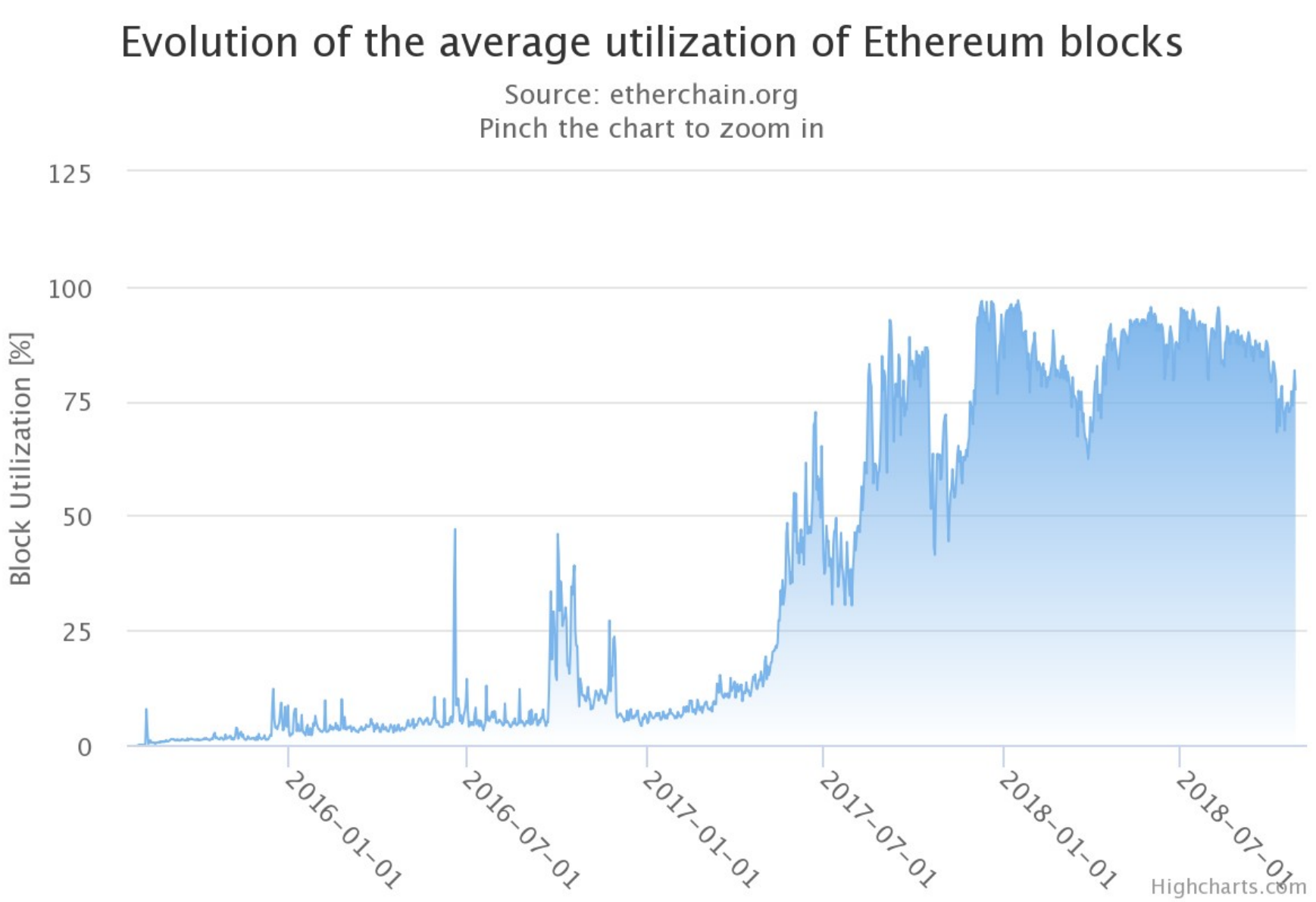
Note: Estimates not valid when multiple transactions are batched from the same address or for transactions sent to addresses with many (e.g. > 100) pending

Top 10 Miners by Blocks Mined: Support for user transactions

Miner	Lowest gas price (gwei)	Weighted avg gas price (gwei)	% of total blocks
0xd4383232c8d1dbe0e03bdfab849871fa17e61807	0	13	1
miningpoolhub	1	11	10
Dwarfpool	1	12	2
Ethermine	1	13	27

Misc Stats (Last 1,500 blocks)

Category	Value
Cheapest Gas Price (gwei)	0
Highest Gas Price (gwei)	14286
Median Gas Price (gwei)	5
Cheapest Transfer Fee	$0.0043


Evolution of the average utilization of Ethereum blocks
Source: etherchain.org
Pinch the chart to zoom in
Block Utilization [%]
125
100
75
50
25
0
2016-01-01
2016-07-01
2017-01-01
2017-07-01
2018-01-01
2018-07-01
Highcharts.com

Performances actuelles

- **block time 15 sec, 4 blocks/min**
- 5959 block/day, 2 174 897 block/year
- **Block Gas limit 8 000 000**
- Daily Gas cap 47 668 965 517
- 76 364 avg gas/tx
- 624 236 tx/day cap
- 433 tx/min
- **7 tx/sec**

En résumé : Pour le calcul...

- L'infrastructure Ethereum est un énorme ordinateur Turing complet distribué
- Ultra tolérant aux pannes
- Trés mal exploité car tous les nœuds exécutent les mêmes instructions avec les même données :-(

En résumé : Pour le stockage...

On est limité par conception à :

- la capacité de stockage des nœuds
- le débit (90 kB / 15 sec => 50 kbits/s)
- le prix du GAS qui fluctue en fonction de l'Ether
- SSTORE : 20.000 GAS/Word -> 640.000 GAS/KB
- 5 GWei / GAS → 0,0032 ETH/KB
- 137.24 $/ETH → 0,439 $USD/KB (25 mars 2019, 11h41)

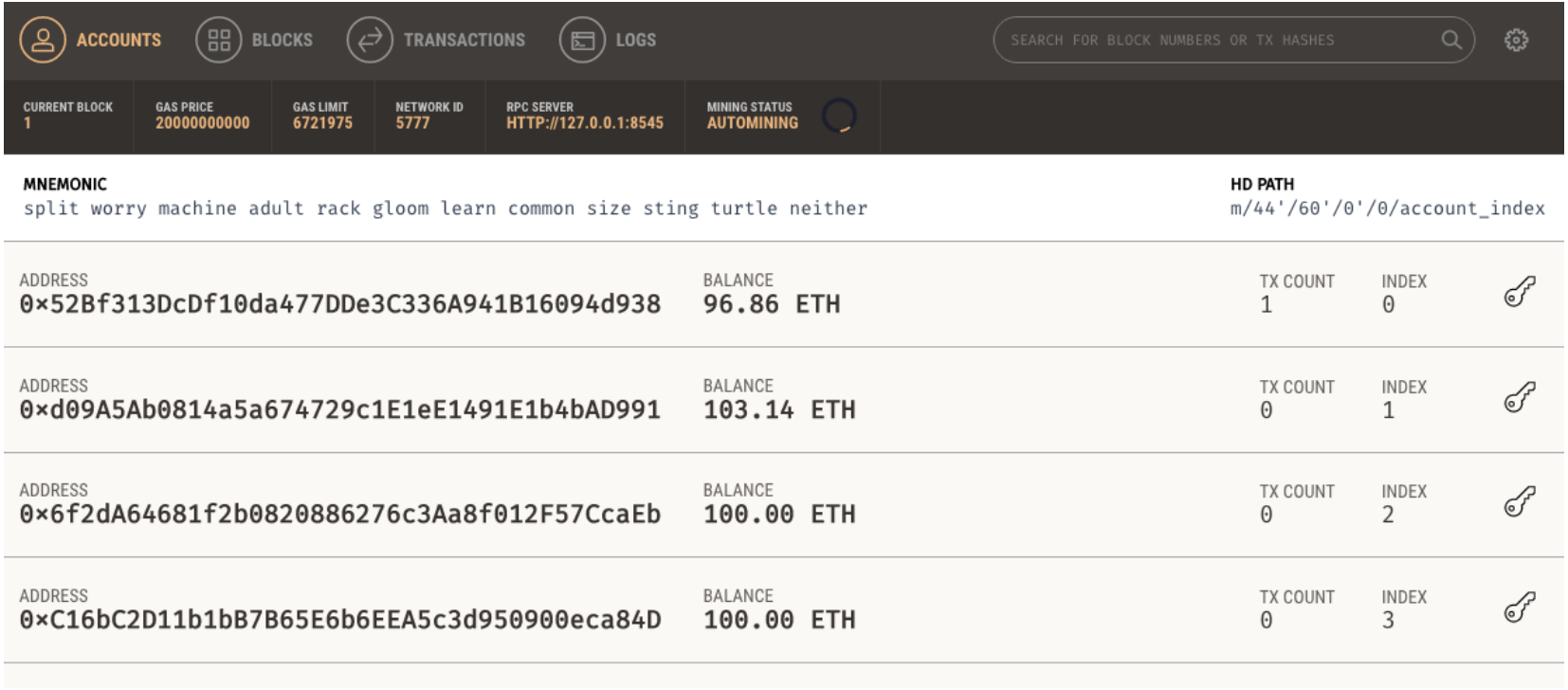
439 000 USD / GBYTES
Temps de transfert : ~45 Heures

(mars 2019 - ~134 USD/ETH, 5 GWei/GAS)

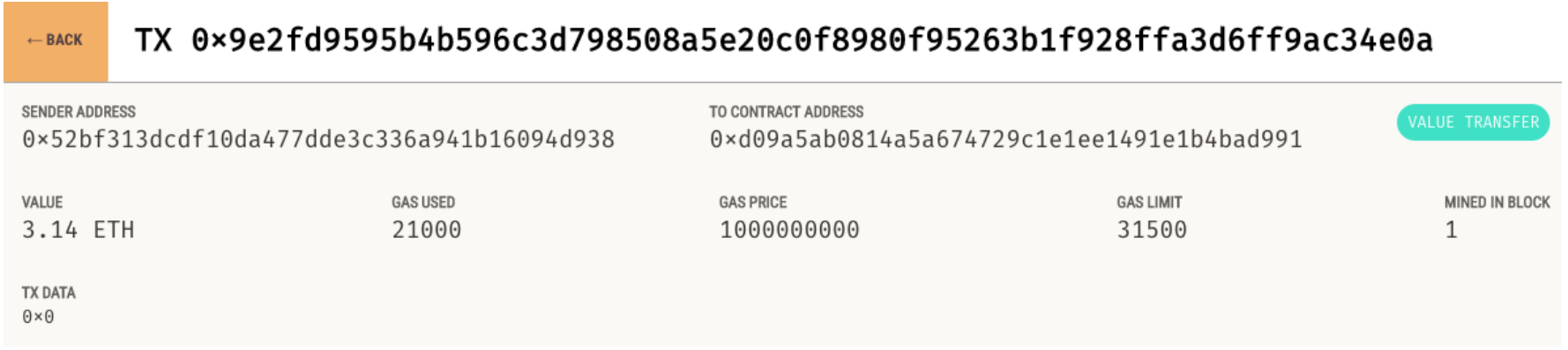
Outils de développement

- Ganache
- Remix
- Truffle

Ganache


ACCOUNTS
BLOCKS
TRANSACTIONS
LOGS
SEARCH FOR BLOCK NUMBERS OR TX HASHES
CURRENT BLOCK 1
GAS PRICE 20000000000
GAS LIMIT 6721975
NETWORK ID 5777
RPC SERVER HTTP://127.0.0.1:8545
MINING STATUS AUTOMINING
MNEMONIC
split worry machine adult rack gloom learn common size sting turtle neither
HD PATH
m/44'/60'/0'/0/account_index
ADDRESS 0x52Bf313DcDf10da477DDe3C336A941B16094d938
BALANCE 96.86 ETH
TX COUNT 1
INDEX 0
ADDRESS 0xd09A5Ab0814a5a674729c1E1eE1491E1b4bAD991
BALANCE 103.14 ETH
TX COUNT 0
INDEX 1
ADDRESS 0x6f2dA64681f2b0820886276c3Aa8f012F57CcaEb
BALANCE 100.00 ETH
TX COUNT 0
INDEX 2
ADDRESS 0xC16bC2D11b1bB7B65E6b6EEA5c3d950900eca84D
BALANCE 100.00 ETH
TX COUNT 0
INDEX 3

Visualisation d’une transaction



Transaction 0x512322644a8bb57fe3fe00977ba462daef2da571a9ee267fc74a969973eae7df

Sponsored: **Azbit.com** - Swiss ICO - Blockchain Banking. Advised by Bitcoin.com founder. **Join the latest Roger Ver's project!**

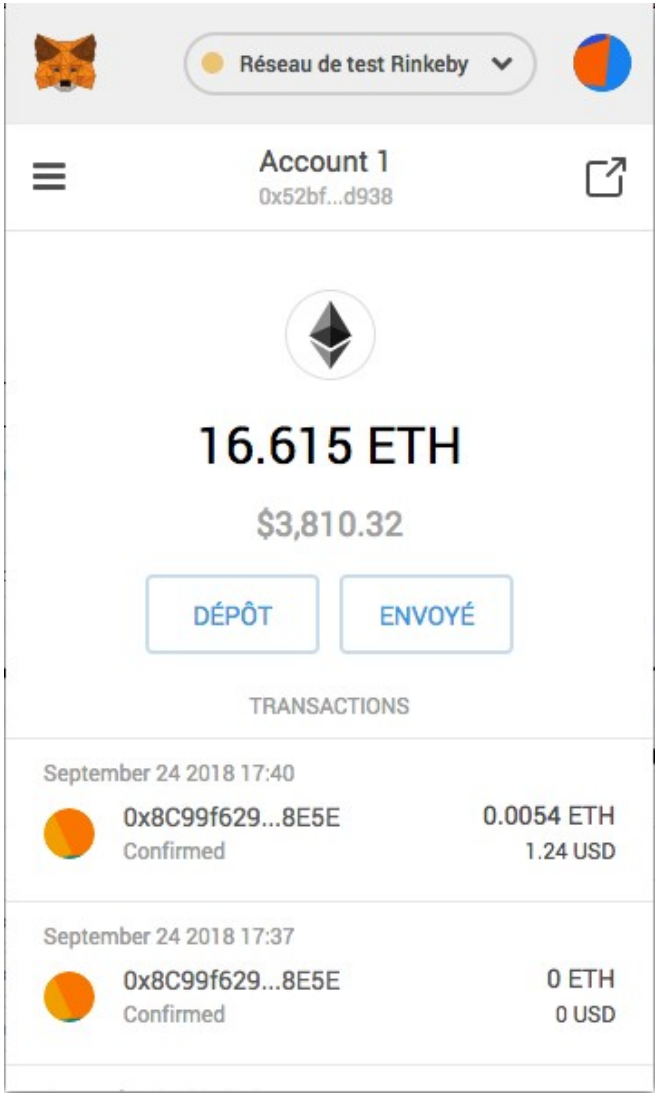
Overview | Comments

Buy | Crypto Loan

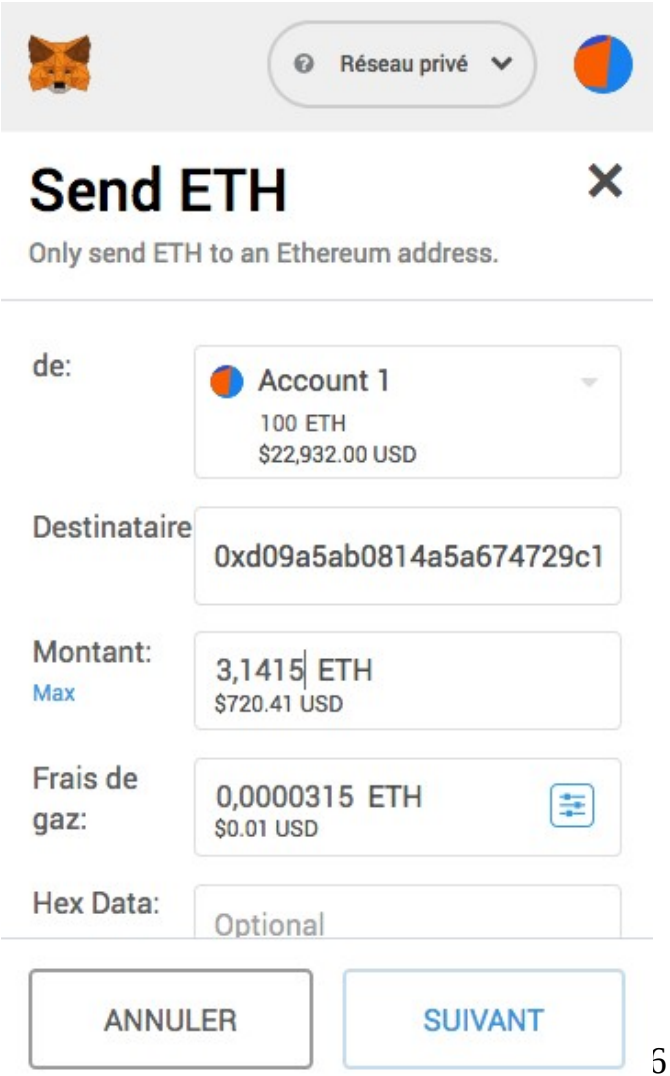
Transaction Information

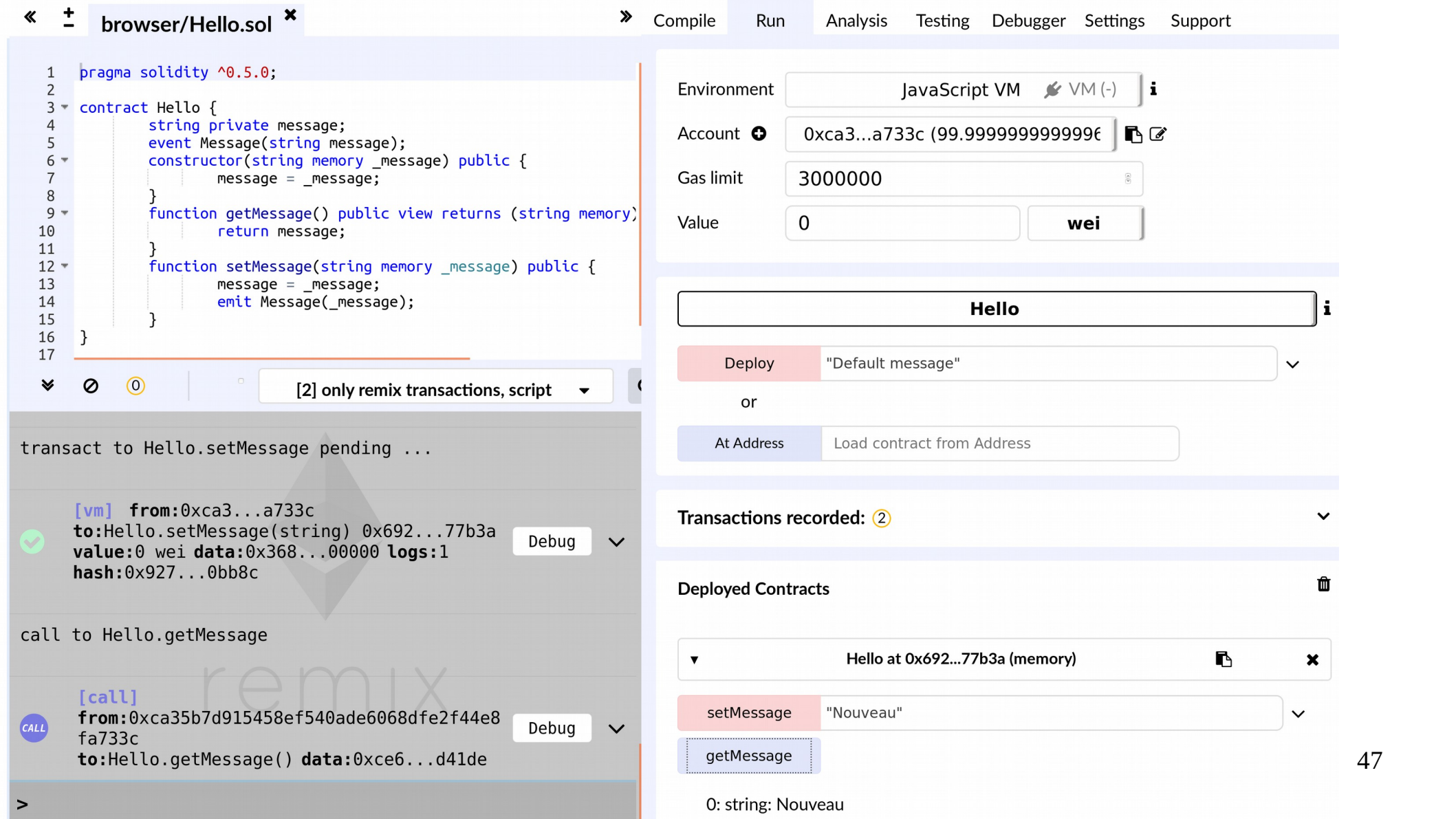
Tools & Utilities

TxHash:	0x512322644a8bb57fe3fe00977ba462daef2da571a9ee267fc74a969973eae7df
TxReceipt Status:	Success
Block Height:	6686011 (1 Block Confirmation)
TimeStamp:	17 secs ago (Nov-11-2018 06:00:36 PM +UTC)
From:	0xc98f4c64c63ce7d10cb7615e0100ffcfc912aba3
To:	0x5d2c3da03b5bc33673b921e8cf5bbae2100e7dc0
Value:	0.054343840721857432 Ether ($11.37)
Gas Limit:	21000
Gas Used By Transaction:	21000 (100%)
Gas Price:	0.000000001401000001 Ether (1.401000001 Gwei)
Actual Tx Cost/Fee:	0.00002942100002 Ether ($0.006158)
Nonce & {Position}:	1 | {123}


Réseau de test Rinkeby
Account 1
0x52bf...d938
16.615 ETH
$3,810.32
DÉPÔT
ENVOYÉ
TRANSACTIONS
September 24 2018 17:40
0x8C99f629...8E5E
Confirmed
0.0054 ETH
1.24 USD
September 24 2018 17:37
0x8C99f629...8E5E
Confirmed
0 ETH
0 USD

Metamask


Réseau privé
Send ETH
Only send ETH to an Ethereum address.
de:
Account 1
100 ETH
$22,932.00 USD
Destinataire
0xd09a5ab0814a5a674729c1
Montant:
Max
3,1415 ETH
$720.41 USD
Frais de gaz:
0,0000315 ETH
$0.01 USD
Hex Data:
Optional
ANNULER
SUIVANT


browser/Hello.sol
Compile
Run
Analysis
Testing
Debugger
Settings
Support
pragma solidity ^0.5.0;
contract Hello {
string private message;
event Message(string message);
constructor(string memory _message) public {
message = _message;
}
function getMessage() public view returns (string memory)
return message;
}
function setMessage(string memory _message) public {
message = _message;
emit Message(_message);
}
}
[2] only remix transactions, script
transact to Hello.setMessage pending ...
[vm] from:0xca3...a733c
to:Hello.setMessage(string) 0x692...77b3a
value:0 wei data:0x368...00000 logs:1
hash:0x927...0bb8c
Debug
call to Hello.getMessage
[call]
from:0xca35b7d915458ef540ade6068dfe2f44e8fa733c
to:Hello.getMessage() data:0xce6...d41de
Debug
Environment
JavaScript VM
VM (-)
Account
0xca3...a733c (99.999999999996
Gas limit
3000000
Value
0
wei
Hello
Deploy
"Default message"
or
At Address
Load contract from Address
Transactions recorded: 2
Deployed Contracts
Hello at 0x692...77b3a (memory)
setMessage
"Nouveau"
getMessage
0: string: Nouveau

Truffle

- Environnement de développement complet.
- `sudo npm install -g truffle`
- `truffle [init|compile|migrate|console|test]`

Test automatique

- ***Mocha*** testing network / ***Chai*** assertion library

```
pragma solidity ^0.4.18;

contract Hello {
    string message = "Default message";

    function getMessage () public view returns (string) {
        return message;
    }
    function setMessage (string _message) public payable {
        message = _message;
    }
}
```

```
const Hello = artifacts.require("Hello");

const dfltMessage = 'Default message';
const newMessage = 'Nouveau';

contract("Hello", accounts => {

        it(`should return \${dfltMessage}`, () =>
        Hello.deployed()
        .then(instance => instance.getMessage())
        .then(message => {
          assert.equal(
            message,
            dfltMessage,
            `The message is not set to \${dfltMessage}`
          );
        })
        );

        it(`should change message to '\${newMessage}'`, () => {
                let meta;
                return Hello.deployed()
                .then(instance => {
                  meta = instance
                  return meta.setMessage(newMessage)
                })
                .then(() => meta.getMessage())
                .then(message => {
                  assert.equal(
                    message,
                    newMessage,
                    "The message has not changed"
                  );
                })
        });
});
```

```
pragma solidity ^0.4.24;

contract EtherTransfer {

  address private owner;

  modifier isOwner {
    require(owner == msg.sender);
    _;
  }
  constructor () public payable {
    owner = msg.sender;
  }
  function () public payable { // callback function
  }
  function getBalance() public view returns (uint) {
    return address(this).balance;
  }
  function withdrawAll() public isOwner {
    uint bal = address(this).balance;
    address(owner).transfer(bal);
  }
  function kill() public {
    selfdestruct(owner); //Destroy the contract
  }
}
```

Simple Smart Contract

- Modifier
- Constructor
- Fallback function
- Selfdestruct

Liens utiles

- https://hackernoon.com/ether-purchase-power-df40a38c5a2f
- OPCODE list + GAS : https://docs.google.com/spreadsheets/d/1m89CVujrQe5LAFJ8-YAUCcNK950dUzMQPMJBxRtGCqs/edit#gid=0
- Ethernodes (28/10/2018 -> 13320 nodes) : https://www.ethernodes.org/network/1
- https://www.etherchain.org/charts/averageBlockUtilization
- https://etherscan.io/
- https://medium.com/coinmonks/storing-on-ethereum-analyzing-the-costs-922d41d6b316

Comparatif énergétique

	Yearly Cost	Energy Used (GJ)
Gold Mining	$105B	475M
Gold Recycling	$40B	25M
Paper Currency and Minting	$28B	39M
Banking System	$1,870B	2,340M
Governments	$27,600B	5,861M
Bitcoin Mining	$4.5B	183M

https://blog.picks.co/pow-is-efficient-aa3d442754d3