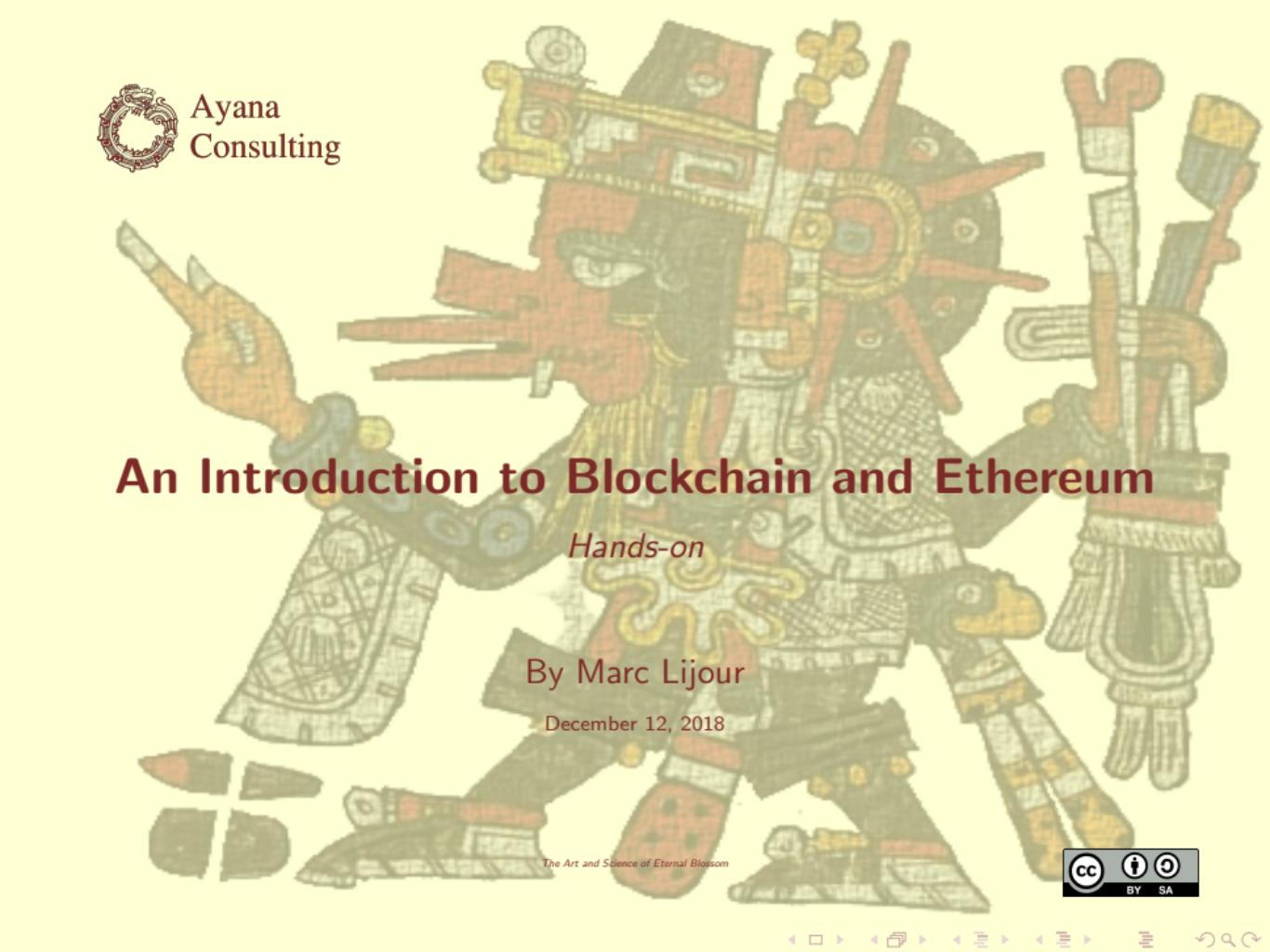




Ayana
Consulting



An Introduction to Blockchain and Ethereum

Hands-on

By Marc Lijour

December 12, 2018

The Art and Science of Eternal Blossom



Who am I?

<https://www.linkedin.com/in/marclijour/>



COLLIDER-X



CONSENSYS



Access these slides

<https://bit.ly/2rvX2HJ>

or find by date:

<https://github.com/marclijour/presentations>



Table of Contents

1 Hands-on Introduction to Crypto

2 Introduction to Smart Contracts

3 Case Studies



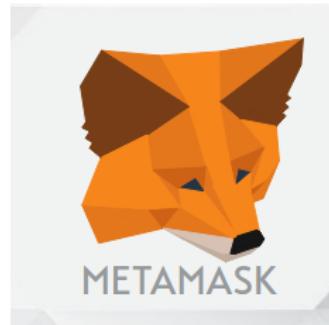
Let's try things out!



Install MetaMask

Follow step by step:

- ① Install the [Chrome/Chromium extension](#)
- ② Watch the [intro on Youtube](#)
- ③ Create an account
- ④ Switch to the Ropsten Testnet
(top-right in MetaMask)
- ⑤ Fill your account with Ether from
<https://faucet.metamask.io>



<https://metamask.io>



Request Ether from the faucet (on the Ropsten network)

Do it several times; then donate 1 ether to the faucet

← → ⌛ 🔒 https://faucet.metamask.io

MetaMask Ether Faucet

faucet

address: 0x81b7e08f65bdf5648606c89998a9cc8164397647
balance: 4094302.54 ether

request 1 ether from faucet

user

address: 0x56e552eb5a9ab277d9eb841f92d473bf0cae7ebf
balance: 1.00 ether

donate to faucet:

1 ether 10 ether 100 ether

transactions



Check the transaction on Metamask

Click on the transaction for a detailed view

The screenshot shows the Metamask extension for a browser. At the top, it says "METAMASK" with a logo, "Ropsten Test Network" (indicated by a red dot), and a network selection dropdown. Below this, the user's balance is shown as "0.9999 ETH" and "\$88.94 USD". There are "DEPOSIT" and "SEND" buttons. On the left, there's a profile section for "Marc 1" with a blue and yellow profile picture, the address "0x56E5...7EBF", and a "DETAILS" button. A large gray box on the left displays the same balance information. Below it, a message says "Don't see your tokens? Click on Add Token to add them to your account." and a "ADD TOKEN" button. The main area shows a transaction history with one entry: "Sent Ether #0 - 12/11/2018 at 23:23" to "0x81b7E0F65Bdf5648606c89998A9CC8164397647" with a value of "1 ETH" and a gas fee of "-\$88.95 USD". The status is "CONFIRMED". Below this, a "Details" section shows the transaction breakdown: Amount 1 ETH, Gas Limit (Units) 21000, Gas Used (Units) 21000, Gas Price (GWEI) 5, Total 1.000105 ETH, and \$88.96 USD. To the right, an "Activity Log" lists three items: "Transaction created with a value of 1 ETH at 23:23 on 3/16/2014.", "Transaction submitted with gas fee of 0 WEI at 23:23 on 3/16/2014.", and "Transaction confirmed at 23:23 on 3/16/2014." There is also a "View on Etherscan" button.



[Check the transaction on Etherscan](#)

ROPSTEN

Etherscan
The Ethereum Block Explorer

ROPSTEN (Revival) TESTNET

Search by Address / Txhash / Block / Token / Ens GO

HOME BLOCKCHAIN TOKEN MISC

Transaction [0xa8a8f32539cb69bd3e506b6527fcc7d44c43cccead09b30b10a9891c99355a3d](#)

Home / Transactions / Tx Info

Overview

Transaction Information Tools & Utilities ▾

[This is a Ropsten Testnet Transaction Only]

TxHash:	0xa8a8f32539cb69bd3e506b6527fcc7d44c43cccead09b30b10a9891c99355a3d
TxReceipt Status:	Success
Block Height:	4610407 (348 Block Confirmations)
TimeStamp:	1 hr 12 mins ago (Dec-12-2018 04:23:14 AM +UTC)
From:	0x56e552eb5a9ab277d9eb841f92d473bf0cae7ebf
To:	0x81b7e08f65bd5648606c89998a9cc8164397647
Value:	1 Ether (\$0.00)
Gas Limit:	21000
Gas Used By Transaction:	21000 (100%)
Gas Price:	0.000000005 Ether (5 Gwei)
Actual Tx Cost/Fee:	0.000105 Ether (\$0.000000)
Nonce & {Position}:	0 {6}
Input Data:	0x



A note about gas price

<https://ethgasstation.info>

ETH Gas Station

Estimates over last 1,500 blocks - Last update: Block 5164391

Change Currency ▾

Std Cost for Transfer \$0.056 | **Gas Price Std (wei)** 3 | **SafeLow Cost for Transfer** \$0.056 | **Gas Price SafeLow (wei)** 3 | **Median Wait (s)** 29 | **Median Wait (blocks)** 2

Gas-Time-Price Estimator: For transactions sent at block: 5164391

Adjust confirmation time

Avg Time (m/s)	4.38
95% Time (m/s)	10.95
Gas Price (Wei)*	3
Tx Fee (Flat)	\$0.056

Gas Used*	21000
Avg Time (blocks)	18.02
95% Time (blocks)	45.05
Tx Fee (ETH)	0.00005

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

Last Block: 5164391

Transaction Count by Gas Price

Confirmation Time by Gas Price

Recommended Gas Prices (based on current network conditions)

Speed	Gas Price (wei)
SafeLow (<30m)	3
Standard (<5m)	3
Fast (<2m)	18

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 nonce conflicts.

Misc Stats (Last 1,500 blocks)



Price of (real) ether: ETH

More information: <https://www.tradingview.com/symbols/ETHUSD/>

ETHUSD Crypto Chart



Wallets

More information: <https://blockgeeks.com/guides/cryptocurrency-wallet-guide/>

Software



Hardware



Paper



Exchanges

- ① Centralized Exchanges (Coinbase, Quadriga, ...)
- ② Decentralized Exchanges

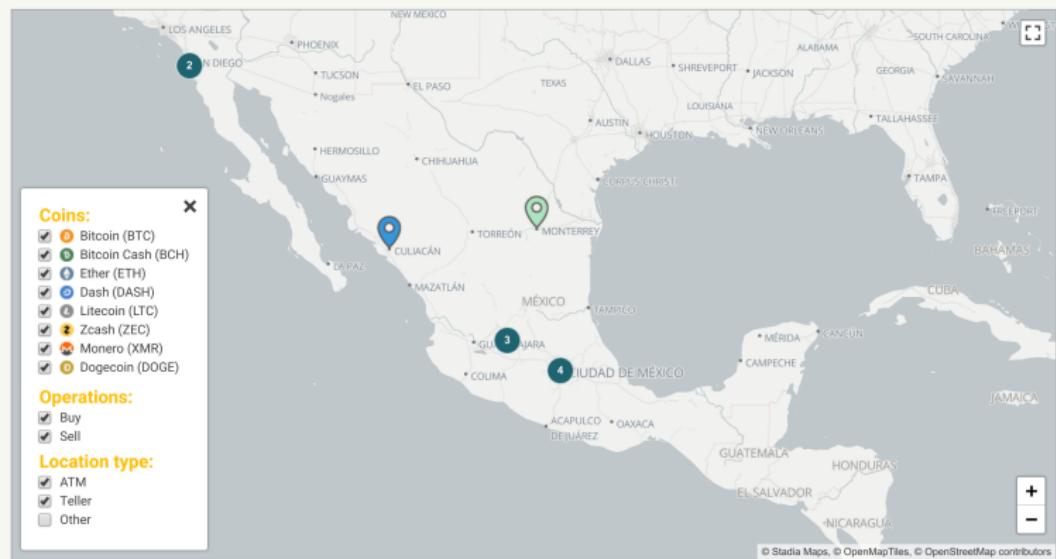


ATMs

More information: <https://coinatmradar.com/country/138/bitcoin-atm-mexico/>

Bitcoin ATMs in Mexico. 🇲🇽

Total number of Bitcoin ATMs / Tellers in Mexico: 11



Other crypto-assets

More information:

<https://www.tradingview.com/markets/cryptocurrencies/prices-all/>

NAME 301 matches	SEARCH	WKT CAP	FD WKT CAP	LAST	AVAIL COINS	TOTAL COINS	TRADED VOL	CHG %
Bitcoin		59.852B	72.158B	3436.1	17.419M	21M	4.712B	0.02%
XRP		12.167B	29.729B	0.29729	40.927B	100B	369.224M	0.17%
Ethereum		9.257B	9.257B	89.23	103.745M	101.745M	1.739B	0.10%
Stellar		2.165B	11.811B	0.11297826	19.165B	104.543B	197.41M	-0.34%
Tether		1.838B	2.554B	0.99000000	1.856B	2.58B	2.97B	-0.10%
Bitcoin Cash		1.729B	2.074B	98.760	17.506M	21M	71.709M	0.92%
EOS		1.692B	1.879B	1.8672	906.245M	1.006B	740.947M	0.44%
Bitcoin SV		1.548B	1.858B	88.45300000	17.505M	21M	59.813M	-2.60%
Litecoin		1.431B	2.028	24.045	59.532M	84M	380.518M	1.14%
TRON		877.572M	1.314B	0.013247	66.247B	99.218B	72.465M	-0.88%
Cardano		772.388M	1.341B	0.029791	25.927B	45B	9.766M	-0.09%
Monero		722.858M	722.858M	43.445	16.638M	16.638M	11.502M	1.38%
NEM		642.663M	642.663M	0.07141	9B	9B	8.23M	-1.51%
Binance Coin		635.009M	926.299M	4.8548350	130.799M	190.799M	20.772M	-2.52%
IOTA		631.509M	631.509M	0.2272	2.78B	2.78B	4.203M	1.07%
Dash		554.925M	1.234B	65.28257000	8.5M	18.9M	160.684M	2.12%
Ethereum Classic		402.997M	402.997M	3.777	106.698M	106.698M	106.521M	0.68%
NEO		387.602M	596.31M	5.96	65M	100M	100.484M	1.51%



Ethereum, Alt-Coins, and ICOs

- ① 94% of the top 100 tokens are built on top of Ethereum
- ② \$5.5 billion raised in 2017
- ③ \$6.5 billion raised in the first quarter of 2018
- ④ Ethereum has the largest developer community
- ⑤ Close to 2,000 dapps (decentralized applications) counted on Ethereum



Crypto as seen by the Regulator (and the taxman)

- ① Cryptocurrency
- ② Utility Token
- ③ Security Token

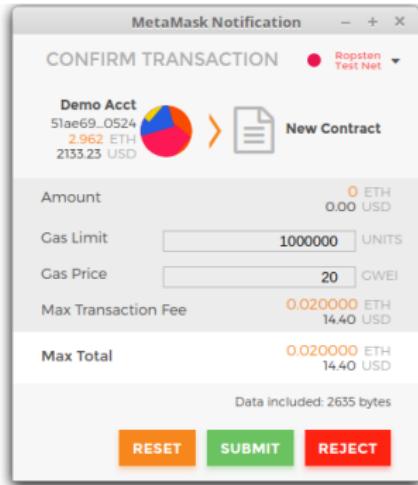


Create your own (ERC-20) token

Create Token

Create Token Contract with the following parameters.

100
Marc's Coin #2
8
MLD
<input type="button" value="Create Token"/>



- ① Use the Token Factory Dapp at <https://tokenfactory.surge.sh/#/factory>
- ② MetaMask will pop up (see picture above)
- ③ Submit the transaction (on the Ropsten Testnet)
- ④ Check your transaction on <https://ropsten.etherscan.io>



Check your Smart Contract

A screenshot of a blockchain transaction status card. At the top, there are two tabs: "SENT" (highlighted in grey) and "TOKENS". Below the tabs, the card displays the following information:

- Icon: A document icon.
- Count: 2
- Date: December 29 2017 04:49
- Description: Contract Published
- Icon: An orange copy icon.
- Value: 0 ETH

- ① Select the “Sent” tab
- ② Check the orange Copy icon (Tx Hash)
- ③ Click on “Contract Published”
- ④ That should bring you to Etherscan (see next page)



Verify the status of your transaction on Etherscan

Transaction Information: note the "To" line with your contract address



Watch your Token



- ① Click on the “Add Token” button
- ② Wait for the next window (picture on the right)
- ③ Copy your contract address (from Etherscan)
- ④ Go back to your Token Factory tab, which should show an UI to interact with your contract or go to the URL:
<https://tokenfactory.surge.sh/#/token/0x...> (replace 0x... by your contract address)
- ⑤ Move coins around
- ⑥ In MetaMask, click on your token to check the tx on Etherscan



Table of Contents

1 Hands-on Introduction to Crypto

2 Introduction to Smart Contracts

3 Case Studies



Remember: Ethereum

Ethereum is a **decentralized platform that runs smart contracts**: applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third party interference.

— <https://ethereum.org>



Coding your first ERC-20 Smart Contract

The screenshot shows the Remix IDE interface. On the left, there's a file tree with a single item: 'browser/ballot.sol'. The main area displays the Solidity code for the 'Ballot' contract. The code defines a struct 'Voter' with fields 'uint weight', 'bool voted', 'uint8 vote', and 'address delegate'. It also defines a struct 'Proposal' with fields 'uint voteCount' and 'address chairperson'. A mapping 'mapping(address => Voter) voters;' is declared, along with a dynamic array 'Proposal[] proposals;'. A constructor function 'Ballot(uint8 _numProposals)' is defined with a note: '/// Create a new ballot with \${_numProposals} different proposals.' On the right side of the interface, there are tabs for 'Compile', 'Run', 'Settings', 'Debugger', 'Analysis', and 'Support'. Below these tabs, there's a button labeled 'Start to compile' with a checkmark for 'Auto compile'. Underneath the tabs, there are buttons for 'Ballot', 'Details', and 'Publish on Swarm'. A message box at the bottom states 'Static Analysis raised 2 warning(s) that require fixing'.

```
pragma solidity ^0.4.0;
contract Ballot {
    struct Voter {
        uint weight;
        bool voted;
        uint8 vote;
        address delegate;
    }
    struct Proposal {
        uint voteCount;
        address chairperson;
    }
    mapping(address => Voter) voters;
    Proposal[] proposals;
    // Create a new ballot with ${_numProposals} different proposals.
    function Ballot(uint8 _numProposals) public {
```

- ① Open the Remix IDE at <https://remix.ethereum.org>
- ② Close the ballot file
- ③ Create a new file named TokenRecipient.sol
- ④ Copy the code from <https://ethereum.org/token> (second white box, under “The Code”, starting with “pragma”)
- ⑤ Switch to the “Run” tab (top-right bar, after Compile)

Reference:

ERC-20 Token Standard



Compiling Successfully

The screenshot shows the Truffle UI interface. On the left, there's a code editor window titled "browser/TokenRecipient.sol" containing Solidity code. On the right, there's a navigation bar with tabs for "Compile", "Run", "Settings", "Debugger", "Analysis", and "Support". Below the tabs, there's a button labeled "Start to compile" with an "Auto compile" checkbox. A dropdown menu shows "TokenERC20" selected. To the right of the dropdown are buttons for "Details" and "Publish on Swarm". A message box indicates "Static Analysis raised 4 warning(s) that require attention". Below this are two green boxes: "TokenERC20" and "tokenRecipient".

```
1 pragma solidity ^0.4.16;
2
3 interface tokenRecipient { function receiveApproval(address _from, uint256 _value, i
4
5+ contract TokenERC20 {
6+     // Public variables of the token
7+     string public name;
8+     string public symbol;
9+     uint8 public decimals = 18;
10+    // 18 decimals is the strongly suggested default, avoid changing it
11+    uint256 public totalSupply;
12+
13+    // This creates an array with all balances
14+    mapping (address => uint256) public balanceOf;
15+    mapping (address => mapping (address => uint256)) public allowance;
16+
17+    // This generates a public event on the blockchain that will notify clients
18+    event Transfer(address indexed from, address indexed to, uint256 value);
19+
20+    // This notifies clients about the amount burnt
21+    event Burn(address indexed from, uint256 value);
22+
23+}
```

- ① Two green boxes should show on the right
- ② TokenERC20 is the name of the contract (class)
- ③ tokenRecipient is the name of the interface
- ④ Switch to the “Run” tab (top right)



Submitting the Smart Contract

The screenshot shows the Truffle UI interface. On the left, there is a code editor window titled "browser/TokenRecipient.sol" containing Solidity code for a TokenERC20 contract. The code defines a public variable `totalSupply` and a mapping of balances for each address. It includes event definitions for approvals, transfers, and burns. On the right, there is a control panel with tabs for "Compile", "Run", "Settings", "Debugger", "Analysis", and "Support". The "Run" tab is selected. Under "Environment", it says "Injected Web3" and shows a dropdown with "0x51a...e0524 (2.948354304538533)". Below that are fields for "Gas limit" (set to 3000000) and "Value" (set to 0). A dropdown menu is open, showing "TokenERC20". Below the dropdown, there are input fields for "Name" ("10, 'Marc 3', 'MLD'") and "Symbol" ("Create"). There is also a "Load contract from Address" field with "At Address" selected. At the bottom, it shows "1 pending transactions" and "0 contract instances".

```
1 pragma solidity ^0.4.16;
2
3 interface TokenRecipient {
4     function receiveApproval(address _from, uint256 _value, string _tokenName, string _tokenSymbol);
5 }
6
7 contract TokenERC20 {
8     // Public variables of the token
9     string public name;
10    string public symbol;
11    uint256 public decimals = 18;
12    // 18 decimals is the strongly suggested default, avoid changing it
13    uint256 public totalSupply;
14
15    // This creates an array with all balances
16    mapping (address => uint256) public balanceOf;
17    mapping (address => mapping (address => uint256)) public allowance;
18
19    // This generates a public event on the blockchain that will notify clients
20    event Transfer(address indexed _from, address indexed _to, uint256 _value);
21
22    // This notifies clients about the amount burnt
23    event Burn(address indexed _from, uint256 _value);
24
25    /**
26     * Constructor function
27     */
28    function TokenERC20(
29        uint256 initialSupply,
30        string tokenName,
31        string tokenSymbol
32    ) public {
33        totalSupply = initialSupply * 10 ** uint256(decimals); // Update total supply
34        balances[_from] = initialSupply; // Give the creator all
35    }
36}
```

- ① Under the dropdown showing “TokenERC20”, add a number (total amount of tokens to issue) and two strings (the latter is the token symbol)
- ② Add enough gas (top right, try 30)
- ③ Click Create and check whether MetaMask needs confirmation



Interacting with the contract

- ① A new interface will pop up on the bottom right corner of the IDE

0 pending transactions

TokenERC20 at 0xca5...ee675 (blockchain)

- totalSupply
- symbol
- name
- decimals
- allowance address , address
- balanceOf address
- transferFrom address _from, address _to,
- burnFrom address _from, uint256 _val
- approve address _spender, uint256 _
- approveAndCall address _spender, uint256 _
- transfer address _to, uint256 _value
- burn uint256 _value



Tools for Developers



Truffle Framework

<http://truffleframework.com>

YOUR ETHEREUM SWISS ARMY KNIFE

Truffle is the most popular development framework for Ethereum with a mission to make your life a whole lot easier.

★ Star 4,734

fork 594

gitter [join chat](#)

INSTALL VIA NPM

```
$ npm install -g truffle
```

Requires NodeJS 5.0+. Works on Linux, macOS, or Windows.

[DOCUMENTATION](#)

[TUTORIALS](#)

Don't know where to start? Get yourself a [Truffle Box!](#)



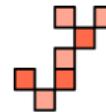
Infura

<http://infura.io>



BLOCKCHAIN BASED

We eliminate the need to install, configure, and maintain costly Ethereum infrastructure.



RELIABLE AND SCALABLE

Our Ferryman™ middleware improves reliability and helps us scale quickly to meet your demand.



DISTRIBUTED STORAGE

Access IPFS seamlessly without the hassle of managing the infrastructure.

POWERFUL AND SECURE

5B+

Requests Per Day

1.6PB

Data Transferred Per Month

9000+

Developers and DApps Served



Mythril

<https://github.com/ConsenSys/mythril>

Mythril is a security analysis tool for smart contracts.

It comes as a Python package that requires a solidity compiler and a C++ compiler.

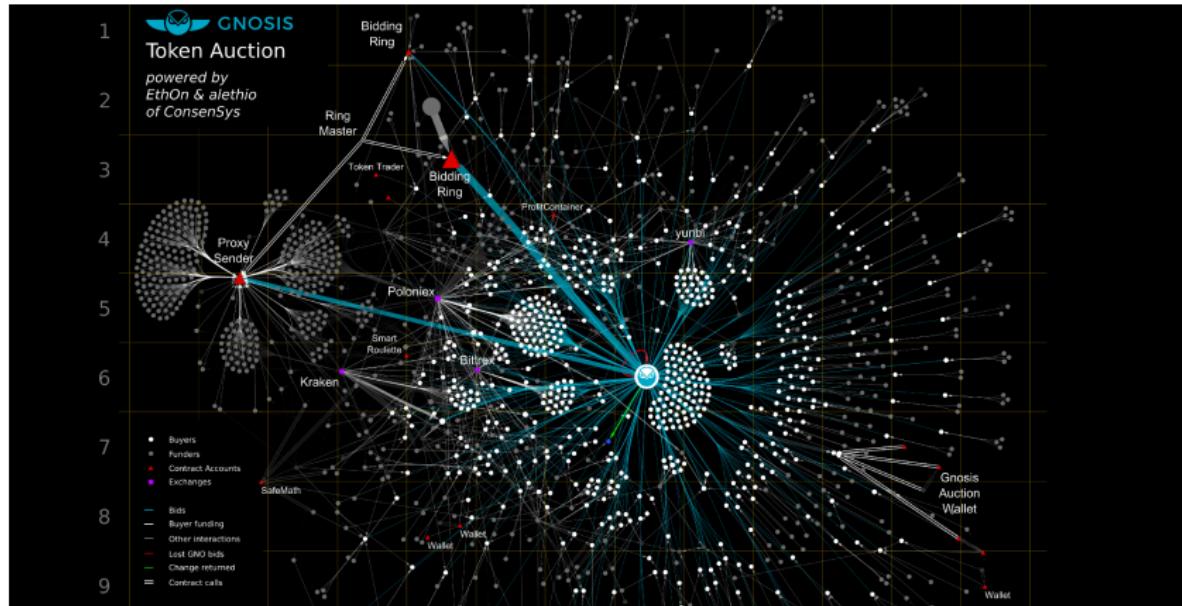
```
$ sudo apt install libssl-dev  
$ sudo apt install gcc g++  
$ sudo add-apt-repository ppa:ethereum/ethereum  
$ sudo apt install solc  
$ sudo pip3 install mythril  
$ myth -x contracts/higherbidder.sol
```

See also <https://hackernoon.com/introducing-mythril-a-framework-for-bug-hunting-on-the-ethereum-blockchain-9dc5588>



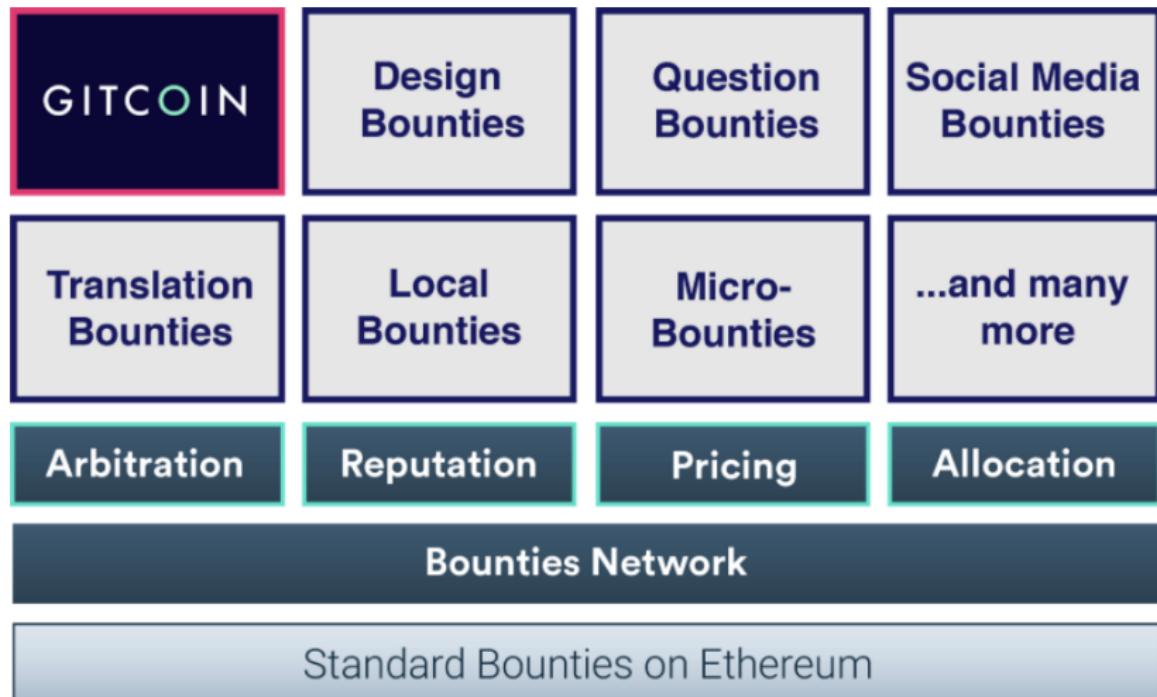
Big Data and Analytics on Ethereum

<https://aleth.io>



Getting paid for your work: The Bounties Network

<https://bounties.network>



Gitcoin (depth-first) and Bounties Network (breadth-first) have integrated!



Gitcoin

<https://gitcoin.co>

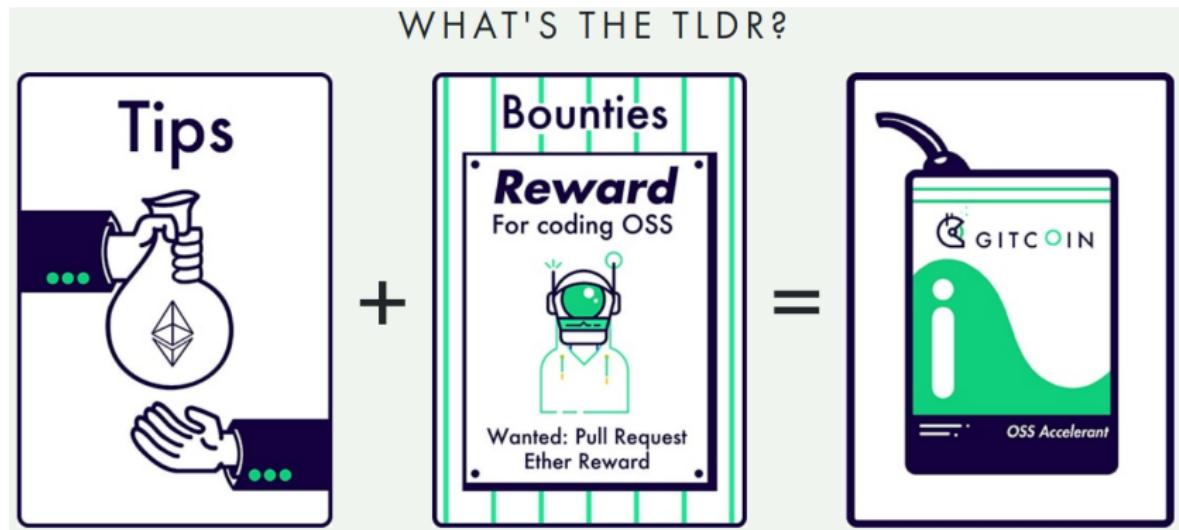


Table of Contents

1 Hands-on Introduction to Crypto

2 Introduction to Smart Contracts

3 Case Studies



Transforming Industries



ConsenSys partnered with a consortium of 15 key industry leaders and leading bank institutions to create a network-based commodities trading platform.



Transforming Industries



UNIONBANK

ConsenSys partnered with UnionBank to develop a closed-loop crypto-cash solution in the Philippines to connecting rural banks to the main financial infrastructure.

KALEIDO

A ConsenSys Business

Try it on aws marketplace



Transforming Industries



mcCarthy
tetraul

ConsenSys has partnered with McCarthy-Tétrault to automate key aspects of the lending process by leveraging smart contract powered loan agreements on the Ethereum blockchain.

In this demo....



- OpenLaw's powerful markup language
- Digital, blockchain-based signatures
- Smart contract execution to a decentralized app ("dApp")



Thank you!

Email: marc@lijour.net

Twitter: [@marclijour](https://twitter.com/marclijour)



References

