

Intro to Ethereum Development and ConsenSys tools

A primer

By Marc Lijour

April 21, 2018



Who am I?

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







Marc Lijour ConsenSys April 21, 2018 2 / 72

Access these slides

https://bit.ly/2qNBiHw

or find by date:

https://github.com/marclijour/presentations





Table of Contents

- Introduction to Ethereum
- 2 Setting a Development Machine
- 3 ConsenSys Stack Overview
- 4 Hands-on Transactions & Smart Contracts
- Developing with Truffle
- 6 Private blockchain experiment





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





5 / 72



Key Milestones:

• (late 2013) Vitalik Buterin describes Ethereum in a paper





- (late 2013) Vitalik Buterin describes Ethereum in a paper
- (Summer 2014) Ethereum raises more than \$14 million in pre-sale





- (late 2013) Vitalik Buterin describes Ethereum in a paper
- (Summer 2014) Ethereum raises more than \$14 million in pre-sale
- (July 30, 2015) Launch of Frontier, initial (beta) version of Ethereum





- (late 2013) Vitalik Buterin describes Ethereum in a paper
- (Summer 2014) Ethereum raises more than \$14 million in pre-sale
- (July 30, 2015) Launch of Frontier, initial (beta) version of Ethereum
- (March 14, 2016) Launch of Homestead, first production release





- (late 2013) Vitalik Buterin describes Ethereum in a paper
- (Summer 2014) Ethereum raises more than \$14 million in pre-sale
- (July 30, 2015) Launch of Frontier, initial (beta) version of Ethereum
- (March 14, 2016) Launch of Homestead, first production release
- (Spring 2016) The DAO





- (late 2013) Vitalik Buterin describes Ethereum in a paper
- (Summer 2014) Ethereum raises more than \$14 million in pre-sale
- (July 30, 2015) Launch of Frontier, initial (beta) version of Ethereum
- (March 14, 2016) Launch of Homestead, first production release
- (Spring 2016) The DAO
- (July 2, 2016) ETH ETC split





- (late 2013) Vitalik Buterin describes Ethereum in a paper
- (Summer 2014) Ethereum raises more than \$14 million in pre-sale
- (July 30, 2015) Launch of Frontier, initial (beta) version of Ethereum
- (March 14, 2016) Launch of Homestead, first production release
- (Spring 2016) The DAO
- (July 2, 2016) ETH ETC split
- (October 16, 2017) Launch of Metropolis (vByzantium) -version 3





Key Milestones:

- (late 2013) Vitalik Buterin describes Ethereum in a paper
- (Summer 2014) Ethereum raises more than \$14 million in pre-sale
- (July 30, 2015) Launch of Frontier, initial (beta) version of Ethereum
- (March 14, 2016) Launch of Homestead, first production release
- (Spring 2016) The DAO
- (July 2, 2016) ETH ETC split
- (October 16, 2017) Launch of Metropolis (vByzantium) –version 3
- (2017) ETH goes from \$7 to more than \$700 (100x increase)

Check the nice infographic (Invezz, 2017).

More information:

- a "prehistory" of the Ethereum protocol (Buterin, 2017).
- the official Ethereum White Paper.



Marc Lijour ConsenSvs April 21, 2018 6 / 72

Store of value



Figure: ETH price (Coindesk, 2017)



7 / 72



Decentralization

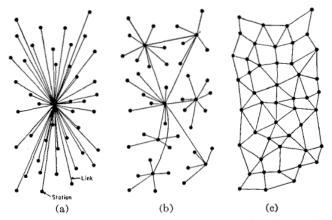


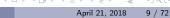
Fig. 1—(a) Centralized. (b) Decentralized. (c) Distributed networks.



Client Types

• Full node





Client Types

- Full node
- Light node





Client Types

- Full node
- Light node
- Something in between (e.g. "fast" for geth)





Full Archive Ethereum node

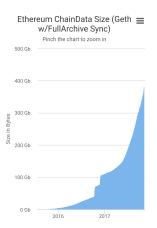


Figure: Miners need a lot of space (Reddit, 2017)



Ethereum vs. Bitcoin

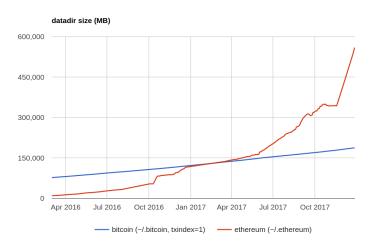


Figure: Disk space used by Geth (fast) vs. Bitcoin (Daniel, 2017)



 Marc Lijour
 ConsenSys
 April 21, 2018
 11 / 72

With Geth --syncmode fast (default mode)

This mode initializes a \sim 20 GB database, then turns in full node.

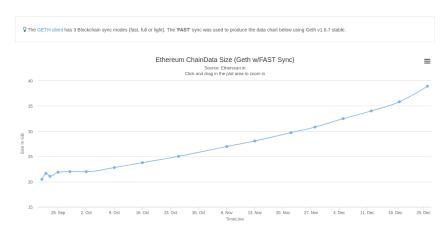


Figure: Disk space used by Geth (in fast mode) (Etherscan, 2017)



Marc Lijour ConsenSys April 21, 2018 12 / 72

Parity allows for continuous state trie pruning

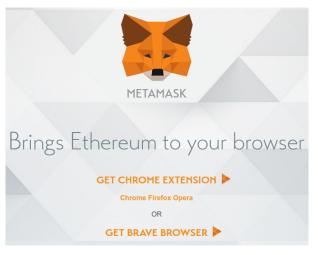
In green, the configuration running as $\it full\ node.$ A light client can fit in $\sim\!5$ MB.

ID	Pruning Mode	Database Configuration	Block Verification	Available Blocks	Available States	Chaindata Size	Parity CLI Flags to use this configuration
0	Archive	+Fat +Trace	Full	All	All	385.000 GB	pruning archivetracing onfat-db on
1	Archive	+Trace	Full	All	All	334.000 GB	pruning archivetracing on
2	Archive		Full	All	All	326.000 GB	pruning archive
3	Fast	+Fat +Trace	Full	All	Recent	37.000 GB	tracing onfat-db on
4	Fast	+Trace	Full	All	Recent	34.000 GB	tracing on
5	Fast		Full	All	Recent	26.000 GB	no-warp
6	Fast	+Warp	Ancient-PoW-Only	All	Recent	25.000 GB	
7	Fast	+Warp -Ancient	No-Ancient	Recent	Recent	5.300 GB	no-ancient-blocks
8	Light		Headers-Only	None	None	0.005 GB	light

Figure: Disk space used by Parity (Afri, 2017)



Metamask



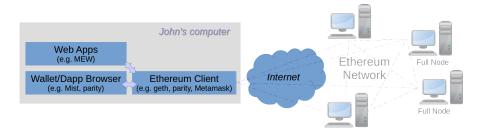
https://metamask.io





Practical Applications

for personal or business use





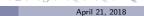


Table of Contents

- Introduction to Ethereum
- Setting a Development Machine
- 3 ConsenSys Stack Overview
- 4 Hands-on Transactions & Smart Contracts
- Developing with Truffle
- 6 Private blockchain experiment



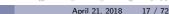


April 21, 2018

Sizing up

- 1) Ethereum Node, Wallet, historical data, Smart Contracts, and Dapps:
 - Linux machine (Ubuntu 16.04 / Linux Mint 18.x –until April 2021)
 - Parity (or Geth)
 - A Solidity compiler
- 2) Developer light setup: (works on ChromeOS)
 - Chrome browser (or Chromium) –any OS
 - Metamask Extension
 - Remix IDE
- 3) Developer Pro setup:
 - truffle





Parity



Figure: The Parity client syncing

- Typical Account Management, multi-sig, hardware support
- Access Dapps directly (e.g. app to create an ERC-20 token)
- Code editor and Solidity compiler for smart contracts
- Fast and reliable (written in Rust)
- Most OS, Docker images; and compliant with JSON-RPC API



Lab 1: set up a full Development Environment

Installing Parity





Installing Parity



 $\label{eq:https://www.youtube.com/watch?v=WNT2O6xyDmM (Windows-based, 16 min)} $$ https://www.youtube.com/watch?v=WNT2O6xyDmM (Windows-based, 16 min) $$ https://www.youtube.com/watch/w$

- Go to https://www.parity.io
- ② Download the relevant binaries, e.g. on Linux:
- Oheck the checksum: \$ md5sum parity_1.7.11_amd64.deb
- Install: \$ sudo dpkg -i parity_1.7.11_amd64.deb
- Oheck the version: \$ parity -v



Run Parity on the Kovan Testnet

Then go to http://localhost:8180 (or http://web3.site if online), and follow the instructions.

- After reading the legal terms and conditions, you can create your first account.
- Click on the top left-most logo (yellow bars) to see the status of your node.
- It may take days to sync!





Try running your first Dapp

Follow the tutorial at https://wiki.parity.io/Deploying-Dapps-to-Parity-Wallet (using chevdor's dapp generator and yeoman)

On Linux Ubuntu, make sure you have npm, and make a soft link to node before running init.sh:

```
$ sudo apt install npm
```





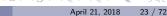
^{\$} sudo ln -s /usr/bin/nodejs /usr/bin/node

^{\$./}init.sh

Lab 1 (b): set up a full Development Environment

Installing Geth





Installing Geth

Instructions (all OSes) at

https://github.com/ethereum/go-ethereum/wiki/Building-Ethereum. Ubuntu/Mint: https://github.com/ethereum/go-ethereum/wiki/Installation-Instructions-for-Ubuntu

```
$ sudo apt-get install software-properties-common
$ sudo add-apt-repository -y ppa:ethereum/ethereum
$ sudo apt-get update
```

Run the first line to install the full suite (geth, bootnode, evm, disasm, rlpdump, ethtest), or the second line for geth only:

```
$ sudo apt-get install ethereum
$ sudo apt-get install geth
```

Create a new account, and you should be ready to run geth:

```
$ geth account new
$ geth
```





Installing a Solidity Compiler

Provided the previous steps were completed:

```
$ sudo apt-get install solc
```

\$ which solc

And in geth, to let it know where solc can be found:

```
$ admin.setSolc("/usr/bin/solc")
```

Now test the code by following the instructions at https://github.com/ethereum/go-ethereum/wiki/Contract-Tutorial





Code Editor



- Vim
- Vim Solidity
- Vim Syntastic





And you still need a wallet

Options:

- Mist Browser (beta) (featured on the right, see also the recent security warning re. Chromium)
- MyEtherWallet (MEW) supports advanced features including hardware wallets



Mist Browser (beta) https://wallet.ethereum.org Try on Chrome vs Firefox



27 / 72



Table of Contents

- Introduction to Ethereum
- Setting a Development Machine
- ConsenSys Stack Overview
- 4 Hands-on Transactions & Smart Contracts
- Developing with Truffle
- 6 Private blockchain experiment





April 21, 2018

ConsenSys Dev Tools in Numbers

https://www.youtube.com/watch?v=WKwR51ANy9E

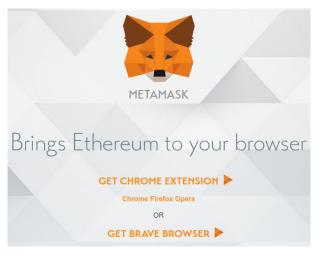






April 21, 2018

Metamask

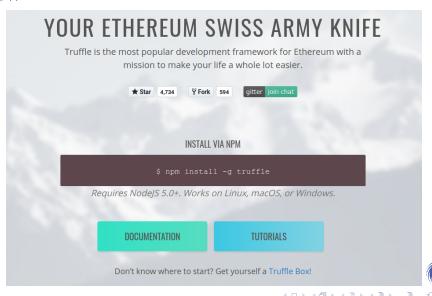


https://metamask.io



Truffle Framework

http://truffleframework.com



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 apt install solc
$ sudo pip3 install mythril
$ myth -x contracts/higherbidder.sol
```

See also https://hackernoon.com/

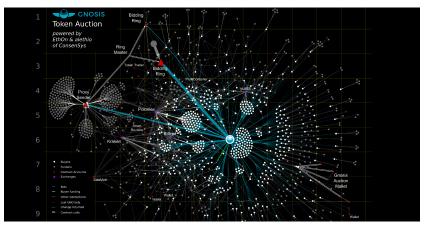
introducing-mythril-a-framework-for-bug-hunting-on-the-ethereum-blockchain-9 dc 5588 and the state of the s



Marc Lijour ConsenSys April 21, 2018 33 / 72

Big Data and Analytics on Ethereum

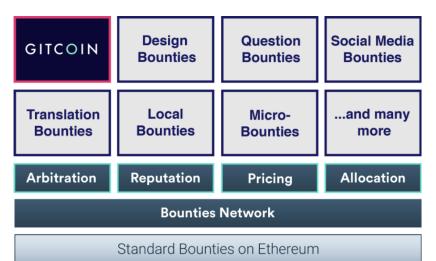
https://aleth.io





Getting paid for your work: The Bounties Network

https://bounties.network



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



 Marc Lijour
 ConsenSys
 April 21, 2018
 35 / 72

4 D > 4 A > 4 B > 4 B >

Gitcoin

https://gitcoin.co

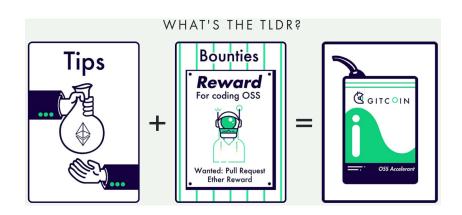






Table of Contents

- Introduction to Ethereum
- 2 Setting a Development Machine
- 3 ConsenSys Stack Overview
- 4 Hands-on Transactions & Smart Contracts
- Developing with Truffle
- 6 Private blockchain experiment





April 21, 2018

Let's have some fun!





Install MetaMask

Follow step by step:

- Install the Chrome/Chromium extension
- Watch the intro on Youtube
- Create an account
- Switch to the Ropsten Testnet (top-left in MetaMask)
- Fill your account with Ether from https://faucet.metamask.io



 $\rm https://metamask.io$



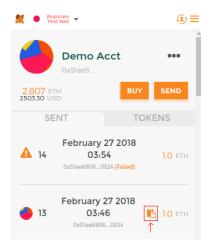
Try sending ETH to yourself with Metamask

- Make sure you're on Ropsten, with some ETH from the faucet
- Click on "Send" and fill:
 - Account: paste your own address (same account)
 - Amount: 1 FTH
 - Transaction data: convert some text in HEX format with https://www.asciitohex.com, remove all spaces and write it with a 0x prefix (e.g. 0x497427732074696d6520746f2072756e)
- Click on "Next"
- Use a gas price > 30 (the higher the faster)
- Confirm the transaction



Check the transaction on Etherscan

Copy the transaction #

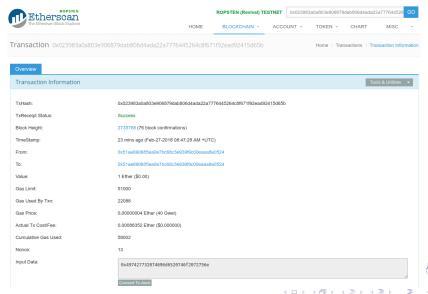






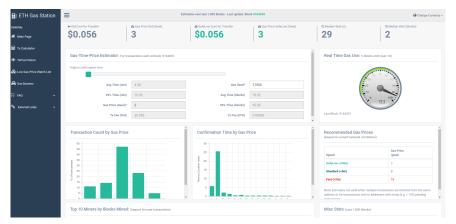
Check the transaction on Etherscan

and click on "Convert to Ascii"



A note about gas price

https://ethgasstation.info







Create your own (ERC-20) token





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



Check your Smart Contract

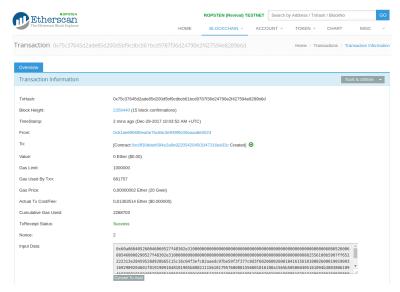


- Select the "Sent" tab
- Check the orange Copy icon (Tx Hash)
- Click on "Contract Published"
- That should bring you to Etherscan (see next page)





Verify the status of your transaction on Etherscan



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



Watch your Token





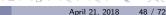
- Olick on the "Add Token" button
- Wait for the next window (picture on the right)
- Oppy your contract address (from Etherscan)
- Go back to your Token Factory tab, which should show an UI to interact with your contract or go to the URL: https://tokenfactory.surge.sh/#/token/0x... (replace 0x... by your contract address)
- Move coins around
- In MetaMask, click on your token to check the tx on Etherscan



Too easy?

Let's code it in Solidity like the pros!





Coding your first ERC-20 Smart Contract



- Open the Remix IDE at https://remix.ethereum.org
- Close the ballot file
- Oreate a new file named TokenRecipient.sol
- Copy the code from https://ethereum.org/token (second white box, under "The Code", starting with "pragma")
- Switch to the "Run" tab (top-right bar, after Compile)

Reference:

ERC-20 Token Standard



<ロト 4回り 4回り 4 更り

Compiling Successfully



- Two green boxes should show on the right
- TokenERC20 is the name of the contract (class)
- tokenRecipient is the name of the interface
- Switch to the "Run" tab (top right)





April 21, 2018

Submitting the Smart Contract



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



4 D > 4 A > 4 B > 4 B >

Interacting with the contract

 A new interface will pop up on the bottom right corner of the IDF



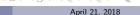




Table of Contents

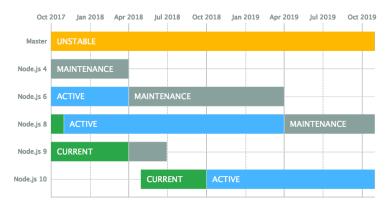
- Introduction to Ethereum
- 2 Setting a Development Machine
- 3 ConsenSys Stack Overview
- 4 Hands-on Transactions & Smart Contracts
- Developing with Truffle
- 6 Private blockchain experiment





Required dependency: node.js > 5

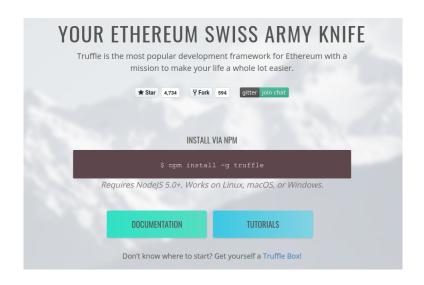
Install version 8 –in LTS maintenance until December 2019. Run a script from https://nodejs.org/en/download/package-manager/







Installing Truffle





Let's build our first Dapp with Truffle

http://truffleframework.com/tutorials/pet-shop



56 / 72



Let's build an ERC20 Token Contract with Truffle

http://truffleframework.com/tutorials/robust-smart-contracts-with-openzeppelin





Table of Contents

- Introduction to Ethereum
- 2 Setting a Development Machine
- 3 ConsenSys Stack Overview
- 4 Hands-on Transactions & Smart Contracts
- Developing with Truffle
- 6 Private blockchain experiment





April 21, 2018

Let's create our own permission-based private Blockchain based on Ethereum!





PoA: Proof of Authority





- PoA: Proof of Authority
- PoA is another type of consensus algorithm (not PoW), with no mining required





- PoA: Proof of Authority
- PoA is another type of consensus algorithm (not PoW), with no mining required
- Less computationally intensive, more secure for small networks, faster





- PoA: Proof of Authority
- PoA is another type of consensus algorithm (not PoW), with no mining required
- Less computationally intensive, more secure for small networks, faster
- The Kovan test network, Hyperledger and Ripple run on a PoA
- Parity supports two PoA consensus algorithm: Aura, and Tendermint (experimental)





Create a PoA chain with Parity

- PoA: Proof of Authority
- PoA is another type of consensus algorithm (not PoW), with no mining required
- Less computationally intensive, more secure for small networks, faster
- The Kovan test network, Hyperledger and Ripple run on a PoA
- Parity supports two PoA consensus algorithm: Aura, and Tendermint (experimental)
- Let's follow Parity's Demo PoA tutorial
- Simple Hands-on at https://github.com/marclijour/parity-poa-tutorial

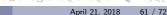




Objectives:

- Setup two connected nodes on one machine (for demo)
- Gain familiarity with Parity (UI and command line)
- Gain a better understanding of diverse types of blockchain (public/private, permissionless/permission-based) and different types of consensus algorithms





Step 1: download the files

This tutorial assumes than you have installed Parity. Instructions are shown for a machine running Linux Ubuntu. The first step consists in cloning the GitHub repo in your machine. You'll run command from within that directory.

\$ git clone https://github.com/marclijour/parity-poa-tutorial.git





Step 2: create nodes and accounts

From the "parity-poa-tutorial" directory, open two terminals and type one line in each:

```
$ parity --config node0.starthere
$ parity --config node1.starthere
```

Open another console and run these scripts:

- \$./create_first_authority_address_on_node0.sh
- \$./create_second_authority_address_on_node1.sh
- \$./create_user__address_on_node0.sh





Step 3: start the chain on PoA

In two separate terminals, restart parity with this new configuration.

```
$ parity --config node0.toml
```





^{\$} parity --config node1.toml

Parity's Demo PoA tutorial - Step 4: setup the Parity UI

Open two different windows or tabs in your browser for node 0 (at http://localhost:8181) and node 1 (at http://localhost:8182).



Restore the accounts as above:

- on node 0: node0 (password = node0), and user (password = user)
- on node 1: node1 (password = node1)



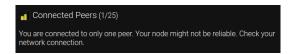


Step 4: connect the nodes with each other

Check the console were you started node 0, and look for the Public Node URL. It should ressemble something like this: enode://<long hash>@<IP Address>:<Port Number>



Go to the Status tab (the leftmost tab) in the Web UI for node 1, and look for the Network Peers section. Click on ADD RESERVED, and copy the URL (including enode://).



Check the console output and the Web UI. Both should acknowledged another peer (1/25 Peers instead of 0/25 Peers).



Step 5: send transactions

Run the following scripts and watch the balance for each account in the Web Uls.

```
$ send_from_user_to_node0_account.sh
$ send_from_user_to_node1_account.sh
```

You can also try in a separate console, where you can read the JSON-formatted response.

```
$check_balance_in_node0_account.sh
$check balance in node1 account.sh
```



67 / 72

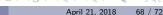


Step 6: add nodes to the network

Run parity with the right chain specification and let other nodes know (by adding them by enode URL). You just need the demo-spec.json file to get started.

\$ parity --chain demo-spec.json





It's the beginning of a rewarding journey...





Next Steps and Recommended Readings

- Starting on Blockchain: key learning resources
- Fairly exhaustive references from Andreessen Horowitz
- Parity Wiki (e.g. Token Deployment)
- Ethereum White Paper and Wiki
- MOOCs: Udemy (Solidity), edX (Hyperledger)
- Building Blockchain Projects: Building decentralized Blockchain applications with Ethereum and Solidity by Narayan Prusty (2017) -check the section on Proof of Authority (PoA)





Thank you!

Email: marc.lijour@consensys.net

Twitter: @marclijour



71 / 72



References

- Afri. (2017, December). The Ethereum-blockchain size will not exceed 1TB anytime soon. Retrieved from https://dev.to/5chdn/the-ethereum-blockchain-size-will-not-exceed-1tb-anytime-soon-58a
- Buterin, V. (2017, September). A Prehistory of the Ethereum Protocol. Retrieved from https://vitalik.ca/general/2017/09/14/prehistory.html
- Coindesk. (2017). Ethereum (ETH) Price. Retrieved from https://www.coindesk.com/ethereum-price/
- Daniel. (2017, December). Retrieved from http://bc.daniel.net.nz
- Etherscan. (2017, December). Ethereum ChainData Size (Geth with fast sync). Retrieved from https://etherscan.io/chart2/chaindatasizefast
- Invezz. (2017). Infographic: the story of Ethereum. Retrieved from https: //cdn4.benzinga.com/files/images/2017/July/05/invezz-eth-history-base.jpg
- Reddit. (2017, November). Ethereum blockchain size...we have a problem. Retrieved from https://www.reddit.com/r/ethtrader/comments/7axn5g/ethereum_blockchain_sizewe_have_a_problem/



Marc Lijour ConsenSys April 21, 2018 72 / 72