By Marc Lijour

October 13, 2018

*One of the largest blockchain companies in the world with 1,000 employees in 30 countries*

CC BY SA

# Who am I?

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

# Access these slides

https://bit.ly/2yh9xuG

or find by date:

https://github.com/marclijour/presentations

# Table of Contents

# The largest Blockchain Cie in the world

We are 1000+ blockchain experts, entrepreneurs, computer scientists, designers, engineers, consultants, and business leaders across 6 continents

# Transforming Industries

# Transforming Industries

# Transforming Industries

# Fair Token Launches

## The Brooklyn Project Framework

**CIVIL** Transparency Scorecard

as of 18 September 2018

**9.2** / 10

Very Good Transparency

| | |
|---|---|
| 1. CONSUMER TOKEN DESIGN | **9/10** |
| 2. PROJECT GOVERNANCE | **10/10** |
| 3. RESPONSIBLE TOKEN DISTRIB. | **10/10** |
| 4. USE OF TOKEN SALE PROCEEDS | **8/10** |
| 5. TOKEN INVENTORY MANAGEMENT | **8/10** |
| 6. CONFLICTS & IMPROPER TRADING | **10/10** |
| 7. TOKEN SAFETY AND AUDIT | **10/10** |
| 8. MARKETING PRACTICES | **8/10** |
| 9. CONSUMER PROTECTION | **9/10** |
| 10. LEGAL COMPLIANCE | **10/10** |

Transparency Grading Scale

| | |
|---|---|
| 1 – 2 | **Red flag**. Non-existent or conflicting disclosures. |
| 3 – 4 | **Poor**. Failure to reveal useful information. |
| 5 – 6 | **Lacking**. Partially complete or unclear disclosures. |
| 7 – 8 | **Good**. Standards met, but room for improvement. |
| 9 - 10 | **Very good**. Complete, clear, consistent disclosures. |

https://framework.thebkp.com/project/CVL/document/2/version/3

TOKEN FOUNDRY

https://tokenfoundry.com/projects/civil

# Table of Contents

# 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

# A short history of Ethereum

Key Milestones:

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

# A short history of Ethereum

Key Milestones:

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

# A short history of Ethereum

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

# A short history of Ethereum

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

# A short history of Ethereum

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

# A short history of Ethereum

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

# A short history of Ethereum

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

# A short history of Ethereum

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 (**ethinfographic**).*

More information:
- a "prehistory" of the Ethereum protocol (**vbuterin2017:prehistory**).
- the official *Ethereum White Paper*.

# 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)

# Disk Space

## Full Archive Ethereum node

Figure: Miners need a lot of space (**reddit:chaindatasize**)

# Disk Space

## Ethereum vs. Bitcoin

Figure: Disk space used by Geth (`fast`) vs. Bitcoin (**daniel:chaindatasize**)

# Disk Space

## With Geth `--syncmode fast` (default mode)

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

Figure: Disk space used by Geth (in fast mode) (**etherscan:chaindatasize**)

# Disk Space

## Parity allows for continuous state trie pruning

In green, the configuration running as *full node*.
A light client can fit in ∼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 archive --tracing on --fat-db on |
| 1 | Archive | +Trace | Full | All | All | 334.000 GB | --pruning archive --tracing on |
| 2 | Archive | | Full | All | All | 326.000 GB | --pruning archive |
| 3 | Fast | +Fat +Trace | Full | All | Recent | 37.000 GB | --tracing on --fat-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:chaindatasize**)

# Metamask

https://metamask.io

# Practical Applications

for personal or business use

# Table of Contents

# 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

# Installing Parity

# Installing Parity

https://www.youtube.com/watch?v=WNT2O6xyDmM (Windows-based, 16 min)

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

# Run Parity on the Kovan Testnet

```
$ parity --light --testnet
2017-12-28 23:38:25  Starting Parity/v1.7.11-stable-a5ed4cf-20171228/x86_64-linux-gnu/rustc1.22.1
2017-12-28 23:38:25  Keys path /home/marc/.local/share/io.parity.ethereum/keys/Kovan
2017-12-28 23:38:25  DB path /home/marc/.local/share/io.parity.ethereum/chains/kovan/db/9bf388941c25ea98
2017-12-28 23:38:25  Path to dapps /home/marc/.local/share/io.parity.ethereum/dapps
2017-12-28 23:38:25  Running in experimental Light Client mode.
...
```

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
```

# 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*

# Table of Contents

# ConsenSys Dev Tools in Numbers

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

# Metamask

https://metamask.io

# 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

# 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

Let's have some fun!

# 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-left in MetaMask)
5. Fill your account with Ether from https://faucet.metamask.io

https://metamask.io

# Try sending ETH to yourself with Metamask

1. Make sure you're on Ropsten, with some ETH from the faucet
2. Click on "Send" and fill:
   - Account: paste your own address (same account)
   - Amount: 1 ETH
   - 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)
3. Click on "Next"
4. Use a gas price $> 30$ (the higher the faster)
5. Confirm the transaction

# Check the transaction on Etherscan

## Copy the transaction #

# Check the transaction on Etherscan

## and click on "Convert to Ascii"

ROPSTEN Etherscan The Ethereum Block Explorer

ROPSTEN (Revival) TESTNET 0x023983a0a803e906879dab806d4ada22a777644526 GO

HOME BLOCKCHAIN ACCOUNT TOKEN CHART MISC

Transaction 0x023983a0a803e906879dab806d4ada22a7776445264c8f671f92ead92415d65b

Home / Transactions / Transaction Information

Overview

Transaction Information

Tools & Utilities

| | |
|---|---|
| TxHash: | 0x023983a0a803e906879dab806d4ada22a7776445264c8f671f92ead92415d65b |
| TxReceipt Status: | Success |
| Block Height: | 2735788 (76 block confirmations) |
| TimeStamp: | 23 mins ago (Feb-27-2018 08:47:28 AM +UTC) |
| From: | 0x51ae690685ea0e7bc68c3e939f9c00eaaa8e0524 |
| To: | 0x51ae690685ea0e7bc68c3e939f9c00eaaa8e0524 |
| Value: | 1 Ether ($0.00) |
| Gas Limit: | 51000 |
| Gas Used By Txn: | 22088 |
| Gas Price: | 0.00000004 Ether (40 Gwei) |
| Actual Tx Cost/Fee: | 0.00088352 Ether ($0.000000) |
| Cumulative Gas Used: | 58002 |
| Nonce: | 13 |
| Input Data: | 0x4974277320746 96d6520746f2072756e |

Convert To Ascii

# A note about gas price

https://ethgasstation.info

ETH Gas Station

Estimates over last 1,500 blocks - Last update: Block 5164359

Change Currency

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) |
|---|---|---|---|---|---|
| $0.056 | 3 | $0.056 | 3 | 29 | 2 |

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

Adjust confirmation time

| | | | |
|---|---|---|---|
| Avg Time (min) | 4.38 | Gas Used* | 21000 |
| 95% Time (min) | 10.95 | Avg Time (blocks) | 18.02 |
| Gas Price (Gwei)* | 3 | 95% Time (blocks) | 45.05 |
| Tx Fee (Fiat) | $0.056 | Tx Fee (ETH) | 0.00006 |

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

Last Block: 5164391

**Transaction Count by Gas Price**

% of transactions

**Confirmation Time by Gas Price**

Time to Confirm (min)

**Recommended Gas Prices**
(based on current network conditions)

| Speed | Gas Price (gwei) |
|---|---|
| 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 transactions

**Top 10 Miners by Blocks Mined:** Support for user transactions

**Misc Stats** (Last 1,500 blocks)

# Create your own (ERC-20) 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

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

ROPSTEN Etherscan
The Ethereum Block Explorer

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

HOME | BLOCKCHAIN | ACCOUNT | TOKEN | CHART | MISC

Transaction 0x75c37645d2ade85d200d5bf9cdbcb61bcd9787f36d24790e2f427594e8289b6d

Home / Transactions / Transaction Information

Overview

Transaction Information — Tools & Utilities

| | |
|---|---|
| TxHash: | 0x75c37645d2ade85d200d5bf9cdbcb61bcd9787f36d24790e2f427594e8289b6d |
| Block Height: | 2358440 (15 block confirmations) |
| TimeStamp: | 2 mins ago (Dec-29-2017 10:03:52 AM +UTC) |
| From: | 0x51ae690685ea0e7bc68c3e939f9c00eaaa8e0524 |
| To: | [Contract 0xc8f30ddeb584e2a8e922054204501f47316eb33c Created] |
| Value: | 0 Ether ($0.00) |
| Gas Limit: | 1000000 |
| Gas Used By Txn: | 681757 |
| Gas Price: | 0.00000002 Ether (20 Gwei) |
| Actual Tx Cost/Fee: | 0.01363514 Ether ($0.000000) |
| Cumulative Gas Used: | 2288703 |
| TxReceipt Status: | Success |
| Nonce: | 2 |
| Input Data: | 0x60a060405260046060527f48302e310000000000000000000000000000000000000000000000000000000000000000006080526006 805460008290527f48302e310000000000000000000000000000000000000000000000000000000882556100b5907ff652 222313e28459528d920b65115c16c04f3efc82aaedc97be59f3f377c0d3f602060026001841615610100026000190190 93 1692909204601f0191909104810190 5b808211156101795760008155600101610 0a1565b505060405161094b3803806109 |

Convert To Ascii

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

Too easy?

Let's code it in Solidity like the pros!

# Coding your first ERC-20 Smart Contract

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

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

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

# Table of Contents

# 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

## 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!

# Let's build our first Dapp with Truffle

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

# Let's build an ERC20 Token Contract with Truffle

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

# Table of Contents

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

# Create a PoA chain with Parity

- PoA: Proof of Authority

# Create a PoA chain with Parity

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

# 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

# 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)

# 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

# Parity's Demo PoA tutorial

Objectives:

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

# Parity's Demo PoA tutorial

## 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
```

# Parity's Demo PoA tutorial

## 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
```

# Parity's Demo PoA tutorial

## 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)

# Parity's Demo PoA tutorial

## 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>`

NETWORK PEERS

ACCEPT NON-RESERVED DROP NON-RESERVED ADD RESERVED REMOVE RESERVED

| | ID | Remote Address | Name | Header (ETH) | Difficulty (ETH) | Capabilities |
|---|---|---|---|---|---|---|
| 1 | 75723705b1b776a9b7 | 192.168.0.16:30301 | Parity/v1.7.11-stable-a5ed4cf-20171228/x86_64-linux-gnu/rustc1.22.1 | ea3995...6d1ed1 | 2.0757224382177246e+40 | eth/62 - eth/63 - par/1 - par/2 - pip/1 |

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://).

Connected Peers (1/25)

You are connected to only one peer. Your node might not be reliable. Check your network connection.

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

# Parity's Demo PoA tutorial

## Step 5: send transactions

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

```
$ 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
```

# Parity's Demo PoA tutorial

## 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

# References