Installation Guide / User Manual

Ethereum Energy Exchange
2x2

1. **Disclaimer: Project created and maintained under linux, specifically linux mint and Ubuntu, and macOS.**
2. **(IMPORTANT) Second Disclaimer: Since cryptocurrency is a relatively new and niche field, it is very volatile to updates and security patches. Therefore, any contracts older than 2 months old will become obsolete for the most part. That being said, almost all of the documentations including the official one are heavily affected by this reasoning as well.**
3. Some keywords:
    a. “{...} = optional”
    b. Ethers = the currency of Ethereum

4. What is cryptocurrency, blockchain, and mining?
    a. Cryptocurrency is a digital currency using cryptography to secure transactions and to keep them anonymous. They rely on a decentralized platform as of right now in which there are no major database or middle man. Everyone operating under this platform contains all of the transaction that occurred hence the blockchain.
    b. Blockchain is essentially a huge ledger or linked list that contains hash pointers. The algorithm is similar to how GitHub operates and inherently they are resistant to change/modification of data. They rely on a peer-to-peer network in which everyone tries to confirm and verifies the transaction (of smart contract in ethereum). To confirm the data, mining occurs and is linked to the blockchain in which everyone downloads to keep track of the history.
    c. Mining is the process to confirm transaction. For the most part, mining uses the computational power of each user to find the encrypted hash pointer. In doing so, each user that taken part in this action receives a reward or ethers (in this case).

5. 3rd Party Applications - Needed to build and run project
    a. Install npm
        i. Linux
            1. curl -sL https://deb.nodesource.com/setup | sudo -E bash -
        ii. Mac
            1. Go to nodejs.org, and click on instal to run through an installer
        iii. Update npm to version 8 and not 9
            1. Npm install npm-8
            2. Npm install -g npm@latest
        iv. Disclaimer: uninstalling npm (regardless of versions) will still have the dependency tree. You must purge everything if uninstalling.
            1. NodeJS has to be version 5 as of right now
            2. Npm has to be version 8 and not 9
    b. Truffle Framework
        i. Linux
            1. Under the assumption of a clean OS reset
            2. Installing and updating packages
                a. Sudo apt-get update && sudo apt-get -y upgrade
                b. Sudo apt-get -y install curl git vim build-essential
            3. Install NodeJS
                a. curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
                b. sudo apt-get install -y nodejs
                c. sudo npm install -g express
            4. Install Truffle
                a. Sudo npm install -g truffle
        ii. macOS
            1. Npm install -g truffle

c. Parity
    i. Installing Rust
        1. Linux
            a. curl https://sh.rustup.rs -sSf | sh
        2. macOS
            a. curl https://sh.rustup.rs -sSf | sh
            b. Disclaimer: clang is required
    ii. One-line installer
        1. bash <(curl https://get.parity.io -Lk)
d. Testrpc - development network
    i. Linux and macOS
        1. npm i ethereumjs-testrpc
e. {Text Editors}
    i. Installing Solidity Plugins
        1. VSCode
            a. Install and open VSCode
            b. Press Ctrl + P and type “ext install”
            c. Type “Solidity” and the first should be it
        2. Sublime Text 3
            a. Install via Package Control
                i. https://packagecontrol.io/packages/Ethereum

6. Understanding Solidity, the language to create smart contracts for Ethereum
    a. Disclaimer: javascript could be used in place of Solidity but that would make creating any kind of smart contract significantly harder.
    b. Solidity is a contract-oriented, high-level language for implementing smart contracts.
        i. It takes aspects of C++, Python, and obviously Javascript
    c. It is statistically typed with multiple inheritance features made to create contracts for nearly every interactions possible.
        i. Voting, auction, medical records, crowdfunding, etc

7. Creating and Starting a project (simple)
    a. Go into a folder you would like to work in
        i. Type “truffle init {nameOfProject}”
        ii. 4 folders and 2 files will be created at the directory - what is it? :
            1. Contracts
                a. Purpose: this contains the smart contract and what kind of application you would like to create
            2. Build
                a. Purpose: this contains the ABI format of the contracts
                    i. This is a javascript aspect that allows the testing network to find what kind of contract you are using
            3. Migrations
                a. Purpose: this contains the contracts you want to submit to the blockchain
            4. Test
                a. Purpose: this contains the test cases you would like to run on the smart contracts
            5. Truffle-config.js and truffle.js
                a. Purpose: contains the network information and how much gas the contract should contain and cost.
        iii. Type either {truffle develop} or {testRPC} in your terminal to test out the contract in the same directory
            1. If testRPC (is one extra step)
                a. Open another terminal and type “truffle console”
            2. In this environment, you are given 10 accounts with addresses and private keys with 100 ethers to use.
            3. Type “truffle compile && truffle migrate --reset” to compile and deploy selected contracts
                a. To select contract to deploy
                    i. Go into “Migration” and create a variable holding the location of the smart contract

ii. Ex. “TokenTransaction” is my smart contract and I want to deploy that.

1. “Var Mig = artifacts.require(“./TokenTransaction.sol”);”
2. “deployer.deploy(Mig);”

4. If no errors appear, then you’re set to use commands to interact with the smart contract.
5. Run web3 commands in order to interact with the contract
6. Type {truffle test} if you have a test (Javascript file in place).

8. More in-depth Example - Using Metacoin.sol or truffle’s default files
   a. What is in it
      i. Build - json files for deployment purposes of the contracts
      ii. Contracts - MetaCoin.sol, ConvertLib.sol, Migration.sol
      iii. Migrations - 1_initial_migrations.js, 2_deploy_contracts.js
      iv. Test - metacoin.js, TestMetaCoin.js
      v. Truffle.js -
   b. What is MetaCoin.sol
      i. It is a coin-like contract that shows you the most basic functionalities or transactions that could be used.
      ii. It teaches how to import contract to each other and call other contract’s functions and to link deployment as well.

c. Example - Running it

i. Using truffle develop as it is more compact than testRPC

```
david@liang ~/Documents/truffle_mock $ truffle develop
Truffle Develop started at http://localhost:9545/

Accounts:
(0) 0x627306090abab3a6e1400e9345bc60c78a8bef57
(1) 0xf17f52151ebef6c7334fad080c5704d77216b732
(2) 0xc5fdf4076b8f3a5357c5e395ab970b5b54098fef
(3) 0x821aea9a577a9b44299b9c15c88cf3087f3b5544
(4) 0x0d1d4e623d10f9fba5db95830f7d3839406c6af2
(5) 0x2932b7a2355d6fecc4b5c0b6bd44cc31df247a2e
(6) 0x2191ef87e392377ec08e7c08eb105ef5448eced5
(7) 0x0f4f2ac550a1b4e2280d04c21cea7ebd822934b5
(8) 0x6330a553fc93768f612722bb8c2ec78ac90b3bbc
(9) 0x5aeda56215b167893e80b4fe645ba6d5bab767de

Mnemonic: candy maple cake sugar pudding cream honey rich smooth crumble sweet treat

truffle(develop)> truffle compile
truffle(develop)> truffle compile --reset
truffle(develop)> truffle migrate --reset
Using network 'develop'.

Running migration: 1 initial migration.js
  Replacing Migrations...
  ... 0x328f720316efabe8cf1fc7bffd00abcb183d1bc15c406ca27a3118efda66215d
  Migrations: 0x8cdaf0cd259887258bc13a92c0a6da92698644c0
Saving successful migration to network...
  ... 0xd7bc86d31bee32fa3988f1c1eabce403a1b5d570340a3a9cdba53a472ee8c956
Saving artifacts...
Running migration: 2 deploy contracts.js
  Deploying ConvertLib...
  ... 0x4fa5daef55338007ca9cd38d011986847b2a9f8bae8df69c55c90e1dfff8472d
  ConvertLib: 0x345ca3e014aaf5dca488057592ee47305d9b3e10
  Linking ConvertLib to MetaCoin
  Deploying MetaCoin...
  ... 0xd163bfa1145157e7b17a57f943ca926182469e81332cd4cfd4b0ff0273fbe911
  MetaCoin: 0xf25186b5081ff5ce73482ad761db0eb0d25abfbf
Saving successful migration to network...
  ... 0x059cf1bbc372b9348ce487de910358801bbbd1c89182853439bec0afaee6c7db
Saving artifacts...
truffle(develop)>
```

ii. As seen, running “truffle develop” will give you 10 accounts. There is a mnemonic phrase that is used for external purposes like running on MetaMask or Parity.

iii. For other reasons, the localhost will be 8545 for most applications but it is 9545 for this particular environment. This allows 3rd party application to use that environment as well.

```
truffle(develop)> contract.then(function(instance) { instance.sendCoin(acc2, 2); });
undefined
truffle(develop)>
```

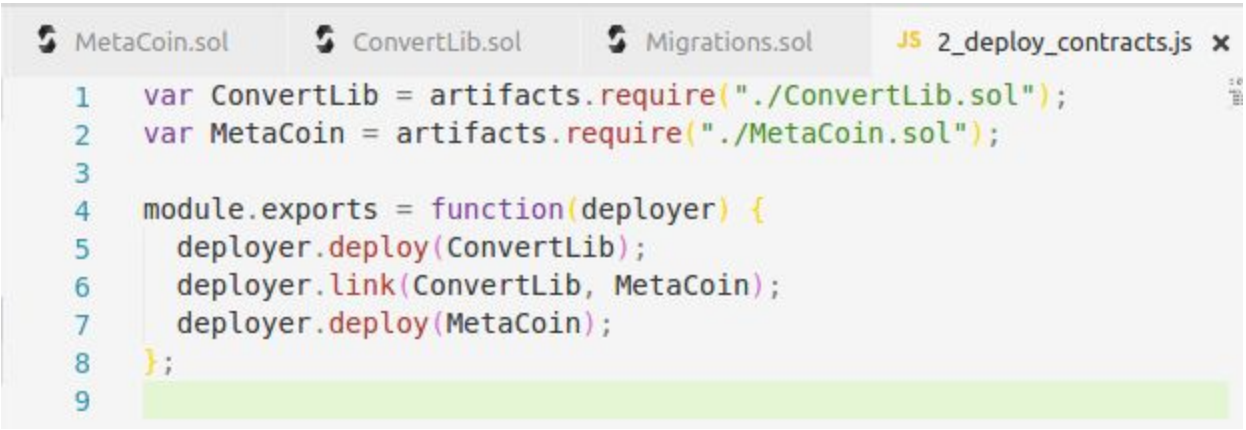
iv. Note: “contract” is “var contract = MetaCoin.deployed()”

v. Ultimately, the results will always be “undefined” unless you link the function call with “console.log”.

1. Sometimes it work and does not work depending on your version.

2. As seen, in order to run the test cases, that command pattern is required and it is also locally scoped.

d. What is deploy? What does it even mean?
   i. Deploy is how you would push your contract to the blockchain which is where mining occurs.
   ii. In the example, the first file “1_initial_migration.js” is always kept there so that you may deploy your contract.
      1. Vice Versa, “Migrations.sol” have to be in your folder as well.
   iii. Ex. In “2_deploy_contracts.js” or whatever you will name it in the future, it contains the files you would like to deploy.

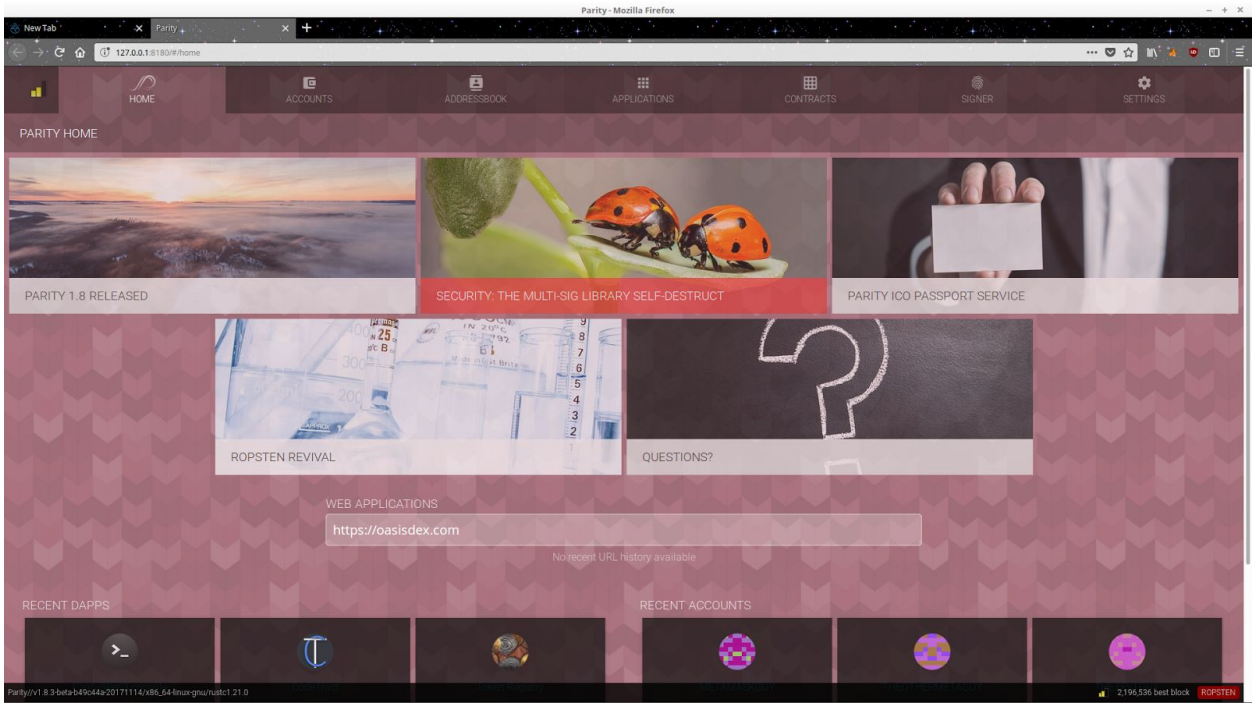


1. By default, it contains “module.exports = function(deployer){}”
2. It mimics compilation in C++ and Java.
3. In this case, there are “ConvertLib.sol” and “MetaCoin.sol” contracts and “MetaCoin” requires the function of “ConvertLib”.
   a. “ConvertLib” has to be deployed first with “MetaCoin” coming in later.
   b. They are obviously linked with “link”.

9. How to use Parity - only for testing purposes as the real blockchain requires actual money
   a. Type “parity ui --chain=ropsten”
      i. “Ui” opens your local browser and opens the parity user interface
         1. Anytime “parity” command is typed, it will begin downloading the global blockchain
      ii. “--chain=ropsten” opens the interface in the ropsten or testnet network.

1. It is a simulator network of the real blockchain but using fake ethers.

iii. Brings this screen up - generally with the network environment at the bottom right corner.

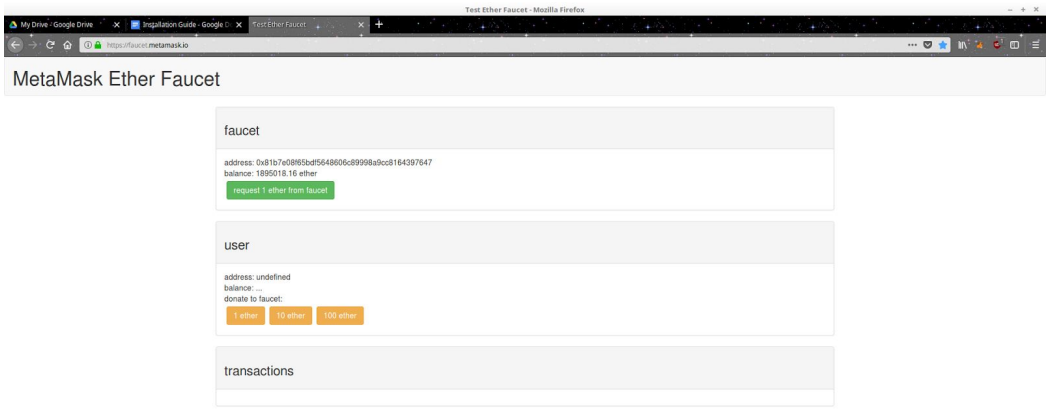


b. How to get ethers in order to deploy my contract to parity and other sites?

i. The only real way besides GPU mining is to create a metamask account.

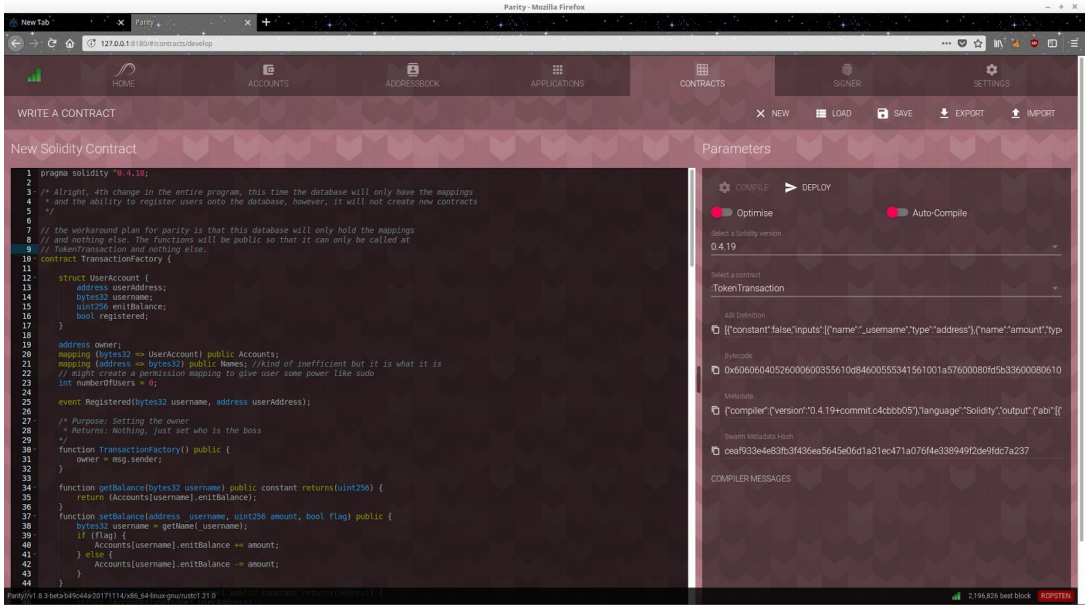
1. Metamask is the same as parity but with a simpler interface to work with and it is an addon on both firefox and chrome
2. Once Metamask account is created, open this site:
    a. https://faucet.metamask.io/
3. In theory, it should have your first account address, and you could now request ether from this site

ii. Such as this:



c. In parity, click on the “contract” tab and click on the “develop” tab which will produce a text editor page

   i. Copy and paste your main contract to the text editor and select your solidity compiler version.



   ii. Then develop and wait for the contract to be mined for confirmation after a few minutes.

   iii. Once deployed, click on “contract” a new contract under the name selected will appear. From there, there will be an interface on how to

interact with the function calls and whom might the addresses will be from and to.