Contents

1. What is Ethereum? 2
2. Why Ethereum? 2
3. How to start? 2
4. Prerequisite 2
Part I 2
Part II 3
5. Quick play with geth: 3
6. Ethereum Wallet: 3
7. Metamask Wallet: 3
8. Build and use Genesis Block: 4
9. Transfer ETH in geth: 4
10. Connect multiple nodes: 4
11. Smart Contract: 5
12. Remix Online Browser: 5
13. Some more with Smart Contract: 5
- Transfer Ether from one account to another using smart contract: 5
- Calling a contract: 5

14. Truffle & testrpc: 6
15. Smart Contract in Truffle: 6
16. Run the sample web page METACOIN: 6
17. Let's build our first web page with Smart Contract: 7
18. A Betting Web page interacting with Smart Contract: 7
Thank You 7

1. What is Ethereum?

Ethereum is an open-source, public, blockchain-based distributed computing platform featuring smart contract (scripting) functionality. .....

2. Why Ethereum?

It is fast growing & popular platform for decentralized applications. A numerous successful applications are already hosted on top of it. You might have seen a recent swarm of ICOs (Initial Coin Offerings) floated.

3. How to start?

You will get enough material online to understand more on various part of this technology. Here we will focus primarily on practical aspect. We are going to start from the basic and will level up slowly to get hands on to understand a simple webpage interacting with Ethereum block-chain in the back ground.

4. Prerequisite

Part I

1- We will work on windows machine. Please note your OS is 32 bit or 64 bit. Based on that you have to install the right compatible application.
   a. Windows 7/10.
   b. Should have admin access to machine.
   c. Google Chrome and Mozilla Firefox installed.
   d. Git : https://git-scm.com/download/win
   e. Metamask google chrome plugin: https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn?hl=en

2- Ethereum command line client **Geth** : https://geth.ethereum.org/downloads/

Install Option#1:- Download and install the installer file for a complete installation.
Install Option#1:- Download the Archive file. It downloads a folder with and exe file 'geth.exe' inside it. One addition step you have to do is to update the windows environment path to point to the location where you are saving the exe file.

For sanity check run '***geth version***' in DOS prompt.

3- Mist or Ethereum wallet : https://github.com/ethereum/mist/releases
Same procedure as above. Either full install or just download and extract the zip file.

4- Solidity Compiler Online: https://remix.ethereum.org

Make sure you are able to open the browser solidity compiler pref

5- Mist or Ethereum wallet : https://github.com/ethereum/mist/releases

Part II

1- Install node pack manager (NPM) - node.js from https://nodejs.org/en/download/
Do a sanity check by typing "***npm –v***" from DOS prompt.

2- Install Truffle and TestRPC from DOS prompt. "***npm install -g ethereumjs-testrpc truffle***".
Do a sanity check by typing "***truffle version***" from DOS prompt.

5. Quick play with geth:

- Open 2 DOS prompt preferably in admin mode.
- Caution: - Do not run '***geth'*** alone without any option. It will try to connect to the main network and eat up your bandwidth, storage space and CPU time. Unless you want to be a node and are aware of the procedures.
- Option#1:- If you want to use 2 windows to see more detail
  - ***geth –-dev : (read 2 '–' dev)*** in cmd window one to invoke the default geth dev client.
  - ***geth attach :*** in 2nd cmd window to open a console and attach to the local client we just started.
- Option#2:- If you want to use only 1 window
  - ***geth –-dev console:*** to invoke the default geth dev client along with console.
- Note the 'datadir:' from the messages you received.
- You may see that couple of folders has been created after our above activities.
- Your accounts are stored in the 'keystore' folder.
- Some Basic commands
  - eth.accounts → To see list of accounts
  - eth.coinbase → Default account
  - eth.blockNumber → Show current block number
  - eth.getBalance(eth.coinbase) → Show the ether balance of account
  - personal.listAccounts → Same result as eth.accounts
  - personal.newAccount("abc") → Create a new account with password "abc"
  - personal.unlockAccount(personal.listAccounts[1]) → To unlock account
  - admin.nodeInfo → Show all node related information
  - miner.start() → To start mining
  - miner.stop() → To stop mining
  - web3/admin/eth/personal → To see what all functions are available in each group.

6. Ethereum Wallet:

1. After running geth node, Open Ethereum Wallet
2. It should automatically connect to our private network.
3. We can see all the accounts, ether mined, block etc..
4. Create an account from the wallet.
5. Transfer ETH from one account to another.
6. Start the miner to complete the transaction.

7. Metamask Wallet:

1. Keep the password and the seed text safe for Metamask.

2. ***admin.startRPC("127.0.0.1", 8545, "*", "web3,net,eth")*** : Start RPC server in our private chain
3. Open Metamask and choose to connect localhost:8545
4. Import the key present in "keystore" folder of our private network.
5. You should be able to see the ETH balance. This is pure test ETH from private network.
6. The main network is the live production network.

8. Build and use Genesis Block:

1. Copy genesis.json in your preferred location. e.g. C:\BlockChain\. Copy from https://github.com/SSPgg/BlockchainWork/blob/master/genesis.json
2. Create a new folder 'Chain1' in your selected location, e.g. C:\BlockChain\Chain1\
3. Navigate to the Chain1 folder in DOS prompt
4. To initialize blockchain use: ***geth --datadir "Chain1" init "genesis.json"***
5. **geth --datadir "Chain1" account new** :- Another way to create a new account before invoking the client.
6. ***geth --datadir "Chain1" console 2> console.log*** : To start the chain and console in same window, this way log the messages in a log file.

9. Transfer ETH in geth:

1. In geth console to unlock an account for longer time use the below web3.personal.unlockAccount("acct#", "pass", 10000000)
2. Transfer Ether from one account to another in console
   a. ***var a = personal.listAccounts*** → Assign the list of addresses to a variable.
   b. ***eth.getBalance(a[1])*** → Check the balance of the receiver account
   c. ***eth.sendTransaction({from:a[0],to:a[1],value:5000000000000000000})*** → Transfer
   d. ***Miner.start()*** to confirm the transaction → Start mining
   e. ***txpool.status*** or ***eth.pendingTransactions***→ To check if any pending transactions
   f. ***miner.stop()*** to stop mining → Stop mining if nothing pending to confirm
   g. ***eth.getBalance(a[1])*** → Check the balance

10. Connect multiple nodes:

- Create 2 folders Node1 & Node2. E.g. under C:\BlockChain folder and initiate them with the genesis file.
  C:\BlockChain>***geth --datadir=.\Node1 init genesis.json***
  C:\BlockChain>***geth --datadir=.\Node2 init genesis.json***
- Start the 2 nodes in separate DOS shell

geth --identity "Node1" --networkid 42 --datadir "./Node1" --nodiscover --rpc --rpcport "8042" --port "30303" --unlock 0 --password ./pass.sec --ipcpath "./Node1/geth.ipc" console 2>.\console1.log

geth --identity "Node2" --networkid 42 --datadir "./Node2" --nodiscover --rpc --rpcport "8043" --port "30304" --unlock 0 --password ./pass.sec --ipcpath "./Node2/geth.ipc" console 2>.\console2.log

- Make sure the nodes have coinbase account and some ETH balance with them.
- ***admin.peers*** → Show if and peers has been added
- ***admin.nodeInfo.enode*** → Get the enode detail of the node.
- ***admin.addPeer("enode")*** → To add another node as peer

- ***eth.blockNumber*** → if connected show same numbers in both nodes
- ***eth.getBlock(eth.blockNumber).miner*** → show the miner of the particular block
- ***eth.getBalance(<addr>)*** → will show the ETH for a particular account

- We can also use static method to add peers.
  - Create a 'static-nodes.json' file listing both enode and place them in both node folders
    File content as example :-> https://github.com/SSPgg/BlockchainWork/blob/master/static-nodes.json
    Replace [::] with the IP address. To get local IP address do ipconfig in DOS prompt.
  - Run the 2 nodes one after another to be added as peer.

- Send some Ether from one acct to another acct present in other node.
  - Check Balance and unlock accts.
  - ***web3.fromWei(eth.getBalance(eth.accounts[1]))*** → Get balance in for of Ether and not Wei.
  - ***eth.sendTransaction({from: eth.coinbase, to: "hard acct",value:web3.toWei(10,"ether")})***
  - Mine from any node to confirm the transaction.
  - ***eth.sendTransaction({from: eth.accounts[1], to: "hard acct",value:web3.toWei(3,"ether")})***

11. Smart Contract:

```
pragma solidity ^0.4.0; //→ Required, tells the base compiler
contract HelloWorld{   //→ contract name
   uint public balance;
   function HelloWorld(){      balance = 1000;   //→ Constructor
   }

   function deposit(uint _value) returns(uint _newValue) { //→ A function
      balance += _value;
      return balance;
   }
}
```

- Deploy in Ethereum wallet to private block chain

12. Remix Online Browser:

- https://remix.ethereum.org → The link for online browser
- Walk thru Compile & Run tab.
- Deploy the hello world smart contract in Remix
- Deploy multiple times and see the difference for them. All contracts run separately.

13. Some more with Smart Contract:

- Transfer Ether from one account to another using smart contract:

https://github.com/SSPgg/BlockchainWork/blob/master/ethTran.sol

- Calling a contract:

Present in META Coin sample contract present in truffle.

14. Truffle & testrpc:

- Go to a folder e.g. – c:\Truffle_UC1\
- Type the command in DOS promt ‘***truffle init webpack***’ – It will set up a sample project folders along with all necessary dependent files.
- After the project initialization, truffle command may not work and throw error pop up. In that case rename the truffle.BAT file in npm folder to a new one, lest say truffle1. The npm folder in my case is ‘C:\Users\ADN\AppData\Roaming\npm’. If truffle is working fine then please ignore this point.
- Run testrpc in one DOS command window “***testrpc –c 5”***
- Run truffle console in another DOS command window “***truffle console”***
- Play with the Ethereum commands we did in geth.

15. Smart Contract in Truffle:

- Let’s create a new contract in contract folder.

```
HelloWeb.sol
pragma solidity ^0.4.2;
contract HelloWeb {
string a;
function setA(string _a) {a = _a; }
function getA() constant returns (string) {return a; }
}
```

- Compile the contract, “***truffle compile***”. See for errors if any.
- Update "2_deploy_contracts" file under "migrations" folder to migrate the new contract into testrpc. It has some example, we need to clone it for our contract. e.g.

```
var HelloWeb = artifacts.require("./HelloWeb.sol");
…..
deployer.deploy(HelloWeb);
```

- Migrate the contract.
  - Make sure testrpc is running in one console, “***testrpc -e 5***”
  - “***truffle migrate***”. See for errors if any. Note down the address of the contract.

- Execute the contract from console.
  - Invoke truffle console by "***truffle console"***.
  - Inside console declare a variable HCon by typing "***var HCon***"
  - Link the variable to our deployed contract by typing the command in console “ ***HelloWeb.deployed().then(function(deployed){HCon=deployed;});***”
  - To execute the function setA and set a value to the variable in the contract, type “***HCon.setA(“Hello Contract Web!!”)”***
  - To get the value we need to execute the getA function, type “***HCon.getA()”.*** You should see the value we have set recently.

16. Run the sample web page METACOIN:

- Give “***npm run dev***” from a command window for web server. Make sure we have testrpc running in another window.
- Open the page http://localhost:8080/ in Mozilla Firefox.

- Do some METACOIN transfer to other addresses.

17. Let’s build our first web page with Smart Contract:

- Copy the index file from the below link to ‘app’ folder in our project ‘https://github.com/SSPgg/BlockchainWork/blob/master/index_HelloWeb.html’
- Change the contract ABI & contract address. When we migrate the contract to blockchain it gets a unique address. We might have saved it after migration.
- Rename the ‘index.html’ file present in ‘app’ folder of our project to ‘index_old.html’
- Rename our file from “index_HelloWeb.html” to “index.html”
- Give “***npm run dev***” from a command window for web server.
- Open the page http://localhost:8080/ in Mozilla Firefox.

18. A Betting Web page interacting with Smart Contract:

- Contract of Betting game - “https://github.com/SSPgg/BlockchainWork/blob/master/BetCon.sol”
- Index file for game – “https://github.com/SSPgg/BlockchainWork/blob/master/index_BetGame.sol”
- Content of “2_deploy_contracts.js”

```
var BetCon = artifacts.require("./BetCon.sol");
module.exports = function(deployer) {
  deployer.deploy(BetCon);
};
```

Thank You

If any comments or suggestion please feel free to contact.

Shakti Patel - shakti.swarup@gmail.com
Ravi Kant Agarwal - ravidilse@gmail.com