Ethereum Development Tutorial

Written by: Pham Hoai Luan

Last revision: May, 2018

How to build Ethereum basic: accounts, contracts, nodes and miners

In previous tutorial, we have successfully completed the sending transaction by Geth client on Ethereum network. Now, we continue comprehending deeply Ethereum protocol by building Ethereum basic. So we have a real scenario about money like: "I want to send some money to my friend who is Mr. B. I could send this money by creating a centralized server, but using an Ethereum money contract comes with some free functionalities: for one, it's a decentralized service and money can be still exchanged even if the original service goes down for any reason. This Ethereum protocol eliminates the scenarios where one single server break-in can result in the loss of money".

To resolve this scenario, I will build the Ethereum money contract. Specifically, I have 3 nodes and the account of node 1 will send some money to the account of node 2 through the Ethereum money contract. Besides, the account of node 3 acts as miner to write transactions on ledger as shown in Figure 1.

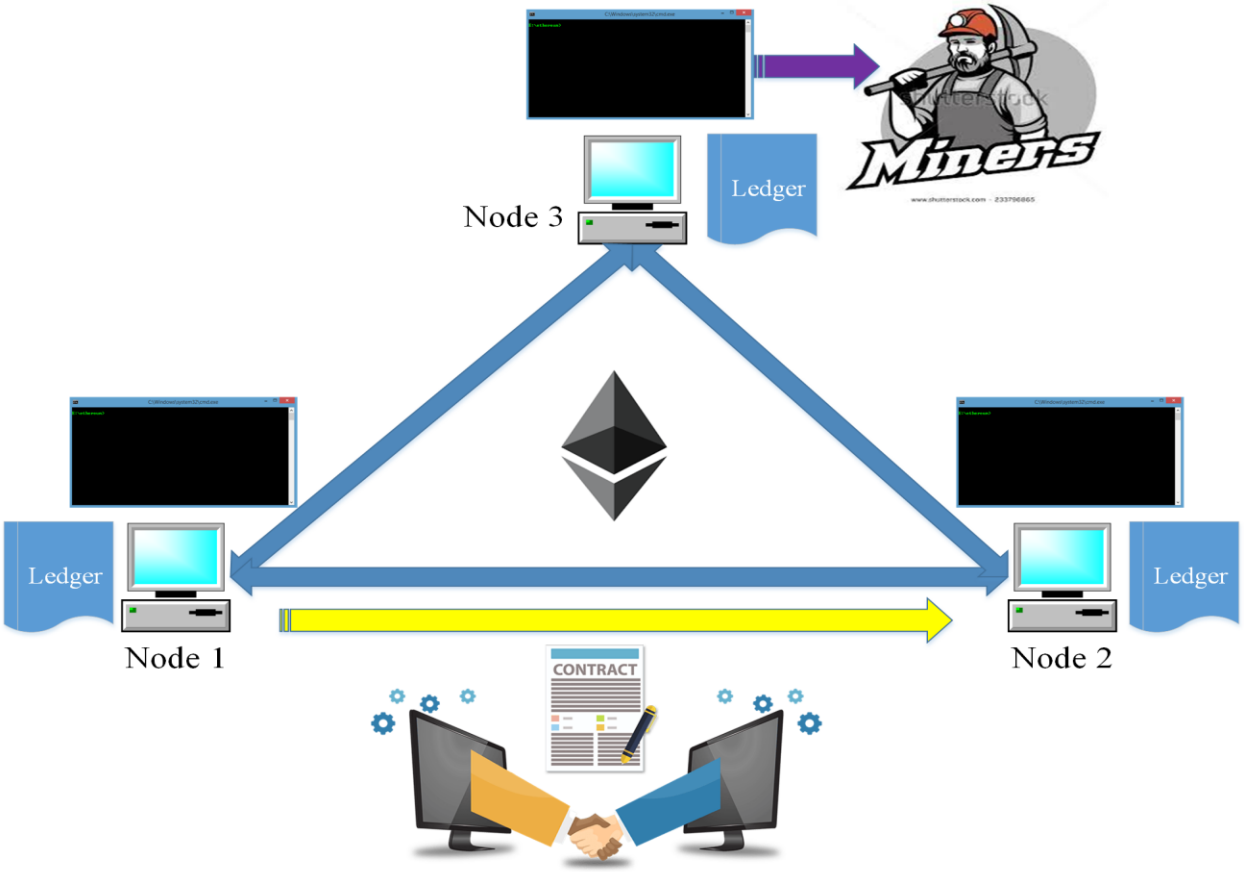

Figure 1. Smart contract Ethereum

Create a network with 3 nodes

I will not explain too much details in this step because it is guided entirely in the previous tutorial. I will go quickly through images.

Starting at node 1

Create a genesis.json file like:

```
{
  "config": {
    "chainId": 1985,
    "homesteadBlock": 0,
    "eip155Block": 0,
    "eip158Block": 0
  },
  "difficulty": "4000",
  "gasLimit": "2100000",
  "alloc": {}
}
```

And put it in an “ethereum_transation” folder as shown in Figure 2.

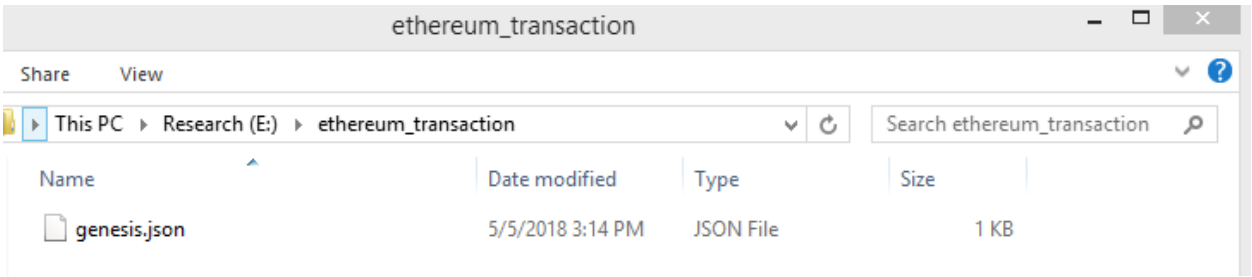

Figure 2. Genesis file

Right-click and choose “Open with Command Prompt as shown in Figure 3.

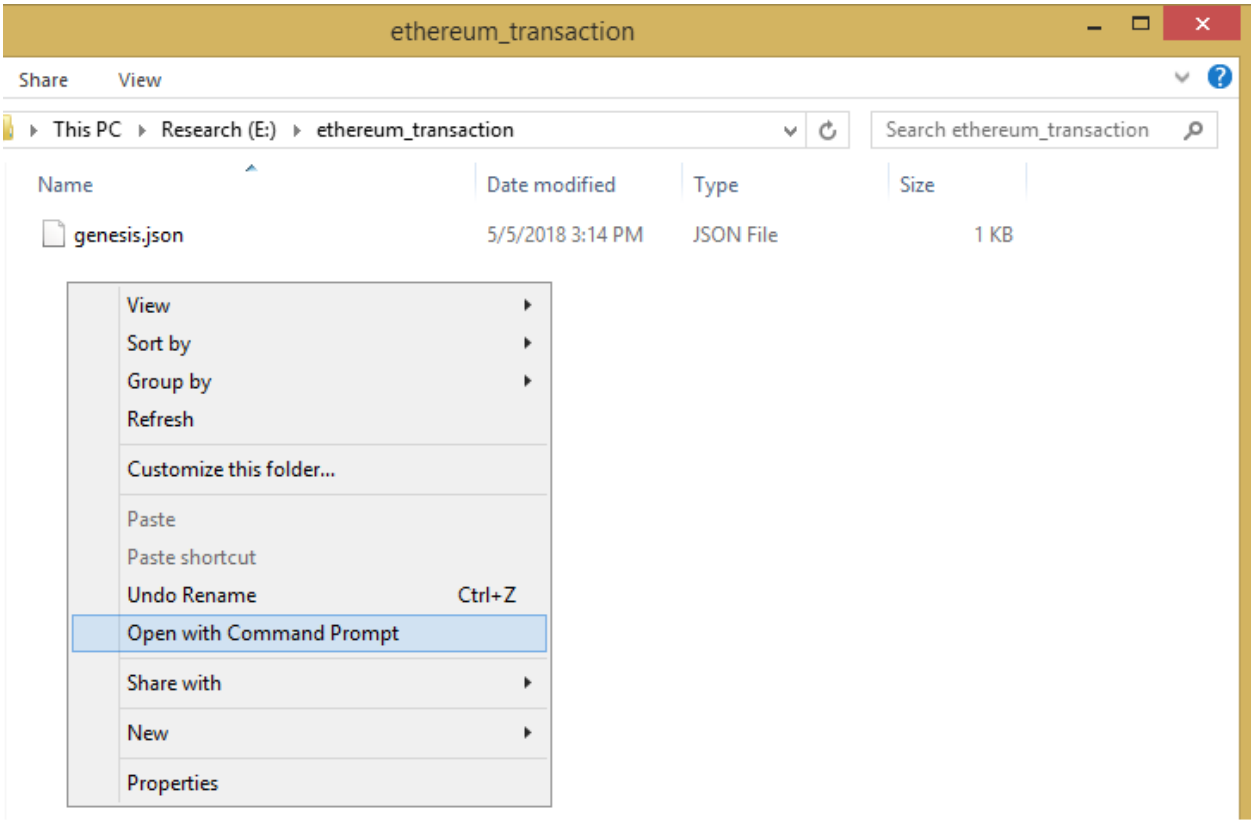

Figure 3. Open with Command Prompt

Then the terminal window will appear as Figure 4.

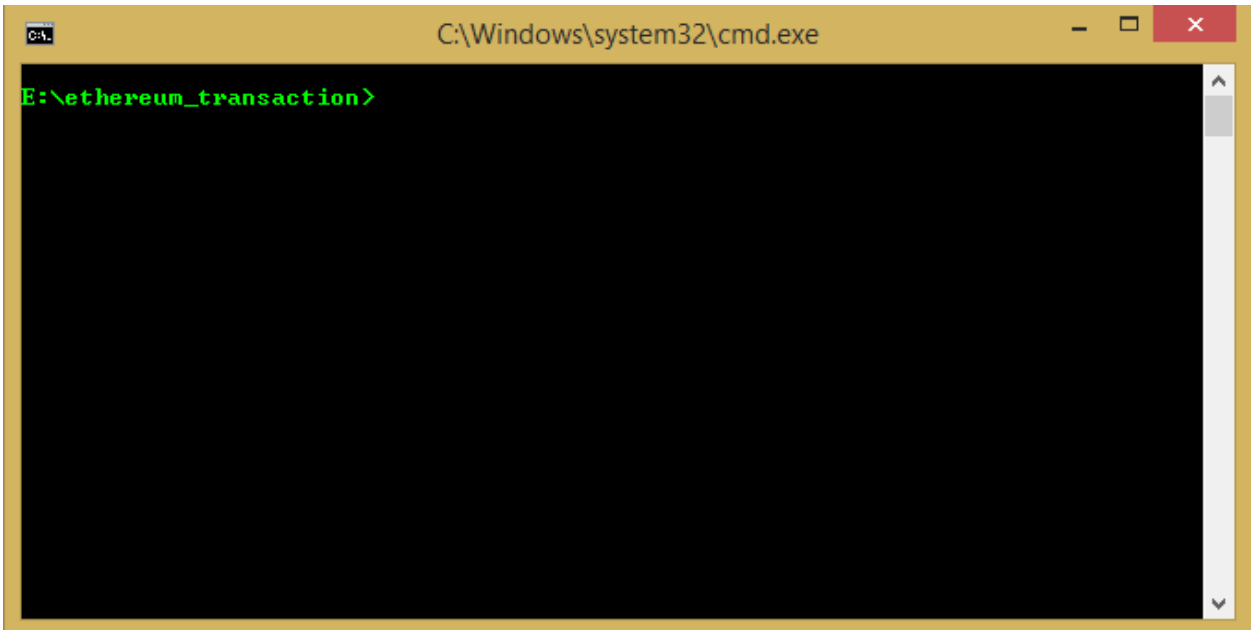

Figure 4. Command Prompt terminal

Type “*geth --datadir "./node01" init "genesis.json"*” in Command Prompt terminal like Figure 5.

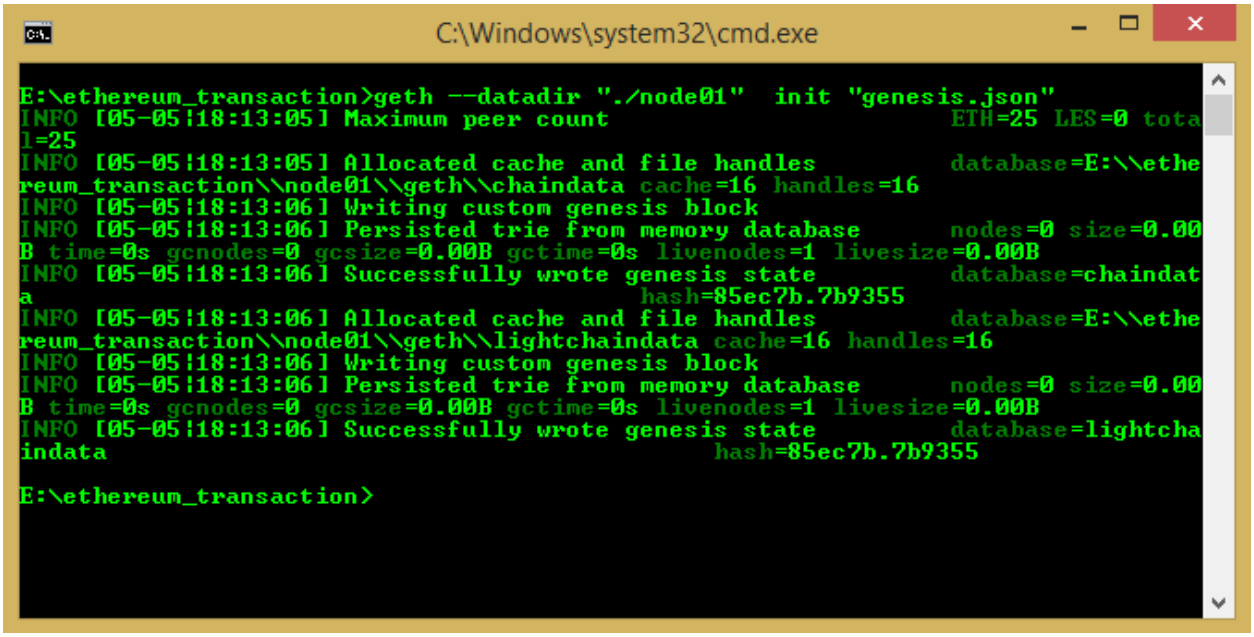

Figure 5. Create genesis block at node 1

Then type “*geth --datadir "./node01" --port 30301 --networkid 1234 --rpc --rpcport "8545" --rpccorsdomain "*" console 2>console.log*” in Command Prompt terminal like Figure 6.

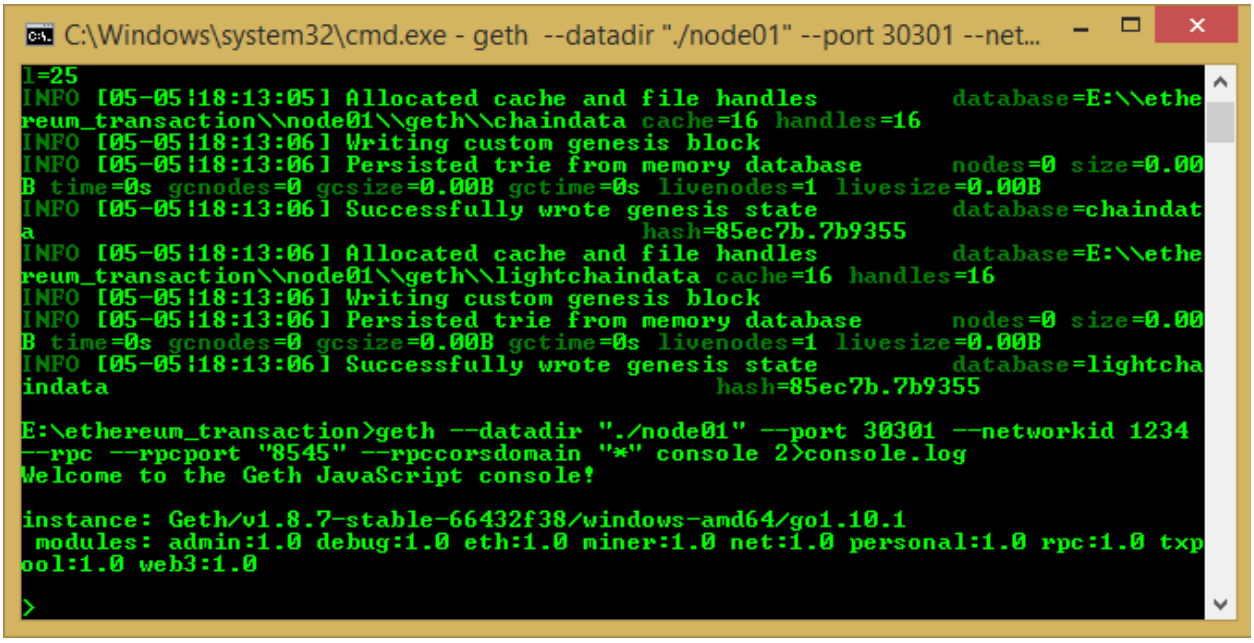

Figure 6. Open Geth console at node 1

Starting at node 2

Right-click and choose “Open with Command Prompt as shown in Figure 3.Then the terminal window will also appear as Figure 4. Type “*geth --datadir "./node02" init "genesis.json"*” in Command Prompt terminal like Figure 7.

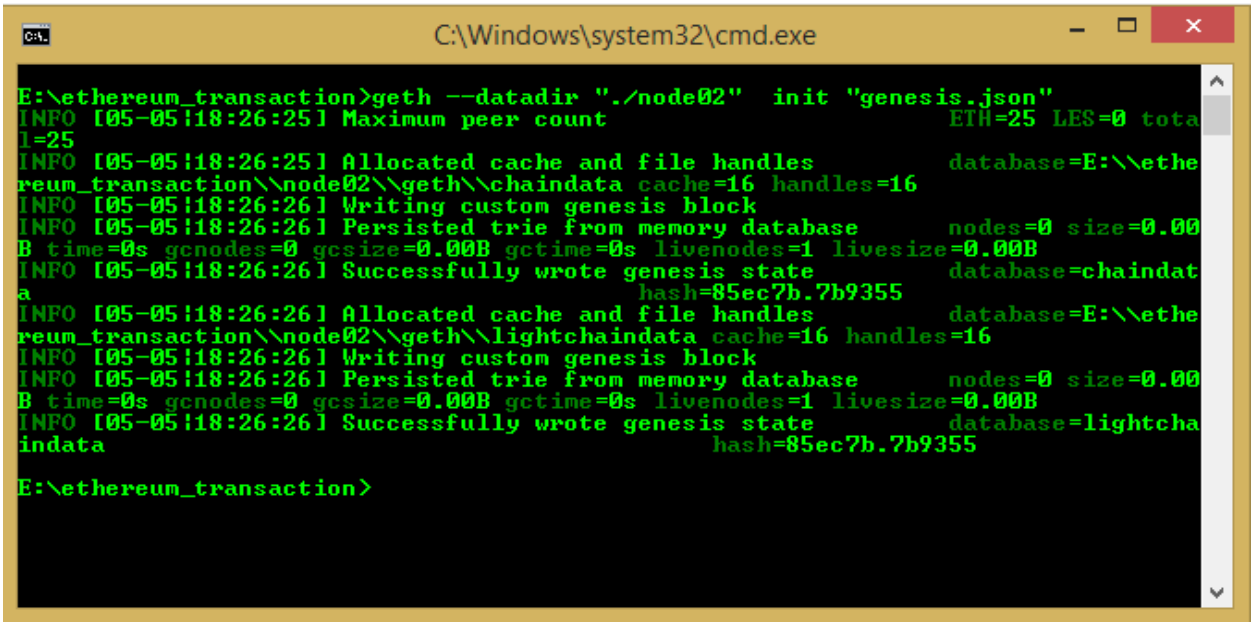

Figure 7. Create genesis block at node 2

Then type “*geth --datadir "./node02" --port 30302 --ipcdisable --networkid 1234 console 2> console2.log*” in Command Prompt terminal like Figure 8.

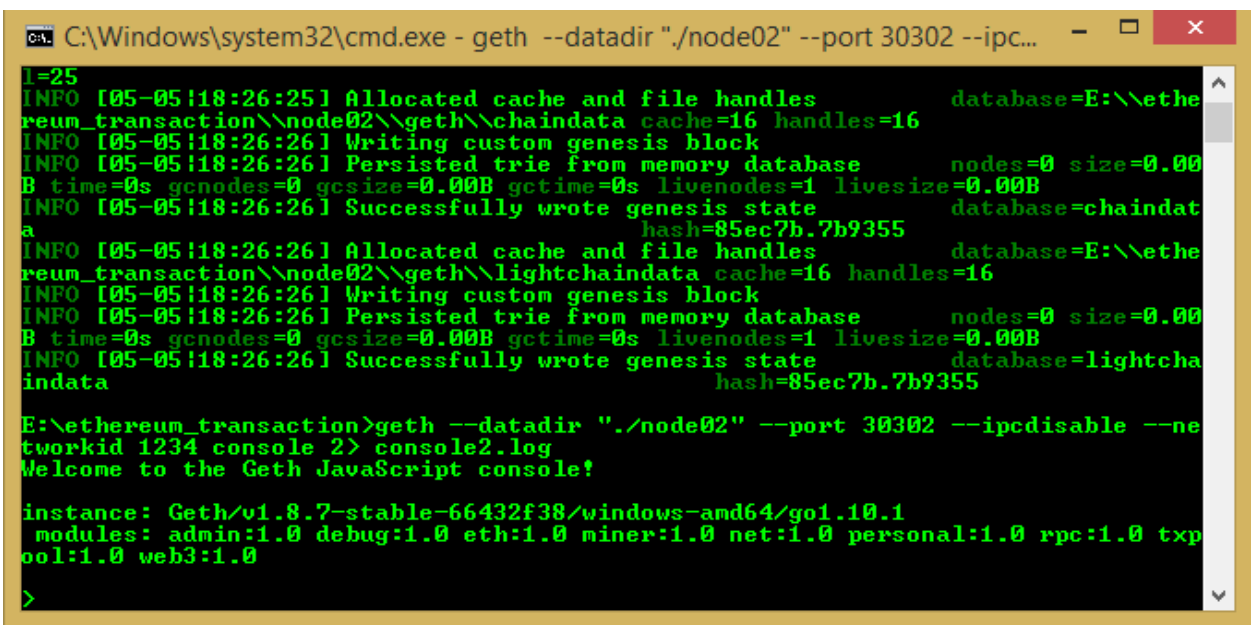

Figure 8. Open Geth console at node 2

Starting at node 3

Right-click and choose “Open with Command Prompt as shown in Figure 3.Then the terminal window will also appear as Figure 4. Type “*geth --datadir "./node03" init "genesis.json"*” in Command Prompt terminal like Figure 9.

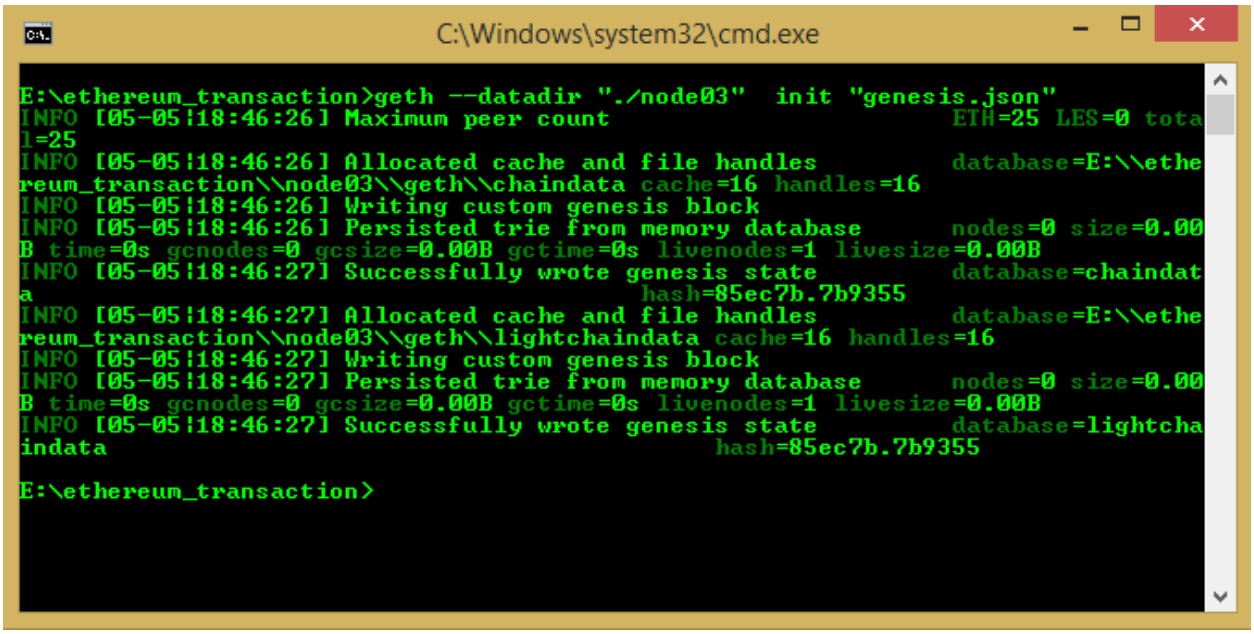

Figure 9. Create genesis block at node 3

Then type “*geth --datadir "./node03" --port 30303 --ipcdisable --networkid 1234 console 2> console3.log*” in Command Prompt terminal like Figure 10.

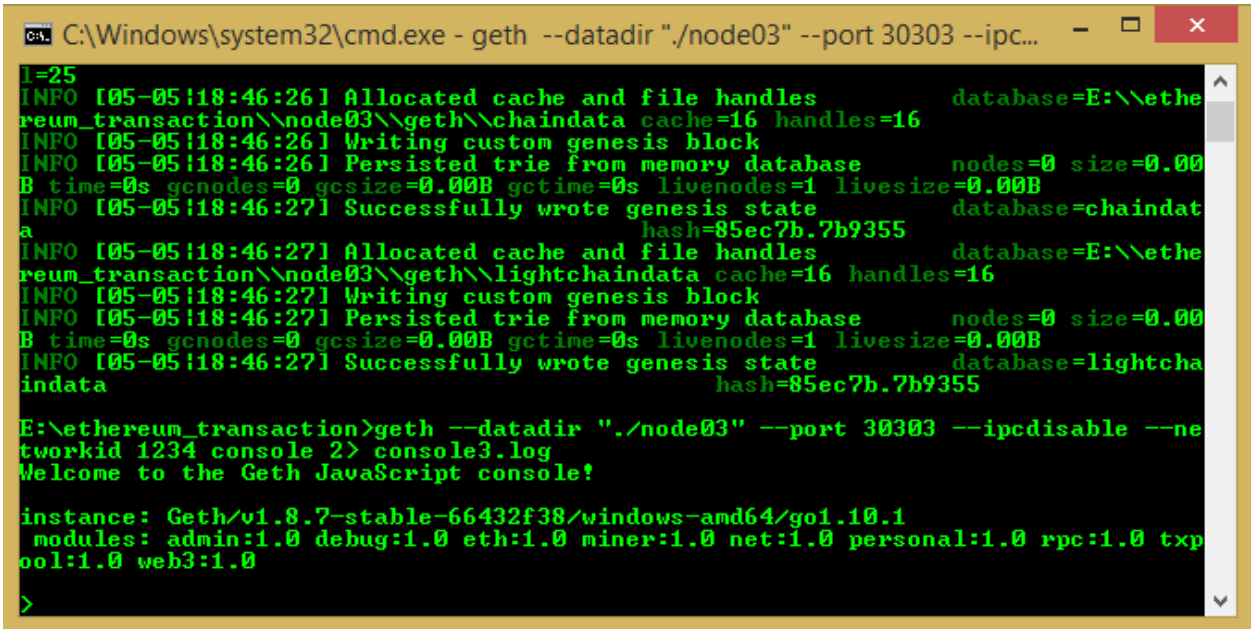

Figure 10. Open Geth console at node 3

Linking 3 nodes

At **node 2,** we type “admin.nodeInfo.enode” like Figure 11.

```
C:\Windows\system32\cmd.exe - geth  --datadir "./node02" --port 30302 --ipc...
INFO [05-06|18:44:44] Writing custom genesis block
INFO [05-06|18:44:44] Persisted trie from memory database      nodes=0 size=0.00
B time=0s gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [05-06|18:44:44] Successfully wrote genesis state         database=chaindat
a                                 hash=85ec7b.7b9355
INFO [05-06|18:44:44] Allocated cache and file handles         database=E:\\ethe
reum_2\\node02\\geth\\lightchaindata cache=16 handles=16
INFO [05-06|18:44:44] Writing custom genesis block
INFO [05-06|18:44:44] Persisted trie from memory database      nodes=0 size=0.00
B time=0s gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [05-06|18:44:44] Successfully wrote genesis state         database=lightcha
indata                                hash=85ec7b.7b9355

E:\ethereum_2>geth --datadir "./node02" --port 30302 --ipcdisable --networkid 12
34 console 2> console2.log
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.7-stable-66432f38/windows-amd64/go1.10.1
 modules: admin:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txp
ool:1.0 web3:1.0

> admin.nodeInfo.enode
"enode://201d687c297e87321aca03235dd362e88f613473a645a74d410eb3bc61fc164ed5d7a3f
0197bfcb74fa6d0343c211fa7480276d393c3f75528e8a504616466ee@[::]:30302"
>
```

Figure 11. Get enode address at node 2

At **node 1,** we type “admin.addPeer(enode address **node 2**) like Figure 12.

```
C:\Windows\system32\cmd.exe - geth  --datadir "./node01" --port 30301 --net...
INFO [05-06|18:44:07] Persisted trie from memory database      nodes=0 size=0.00
B time=0s gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [05-06|18:44:07] Successfully wrote genesis state         database=chaindat
a                                 hash=85ec7b.7b9355
INFO [05-06|18:44:07] Allocated cache and file handles         database=E:\\ethe
reum_2\\node01\\geth\\lightchaindata cache=16 handles=16
INFO [05-06|18:44:07] Writing custom genesis block
INFO [05-06|18:44:07] Persisted trie from memory database      nodes=0 size=0.00
B time=0s gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [05-06|18:44:07] Successfully wrote genesis state         database=lightcha
indata                                hash=85ec7b.7b9355

E:\ethereum_2>geth --datadir "./node01" --port 30301 --networkid 1234 --rpc --rp
cport "8545" --rpccorsdomain "*" console 2>console.log
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.7-stable-66432f38/windows-amd64/go1.10.1
 modules: admin:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txp
ool:1.0 web3:1.0

> admin.addPeer("enode://201d687c297e87321aca03235dd362e88f613473a645a74d410eb3b
c61fc164ed5d7a3f0197bfcb74fa6d0343c211fa7480276d393c3f75528e8a504616466ee@[::]:3
0302")
true
>
```

Figure 12. Add node 2 at node 1

At **node 3,** we type “admin.nodeInfo.enode” like Figure 13.

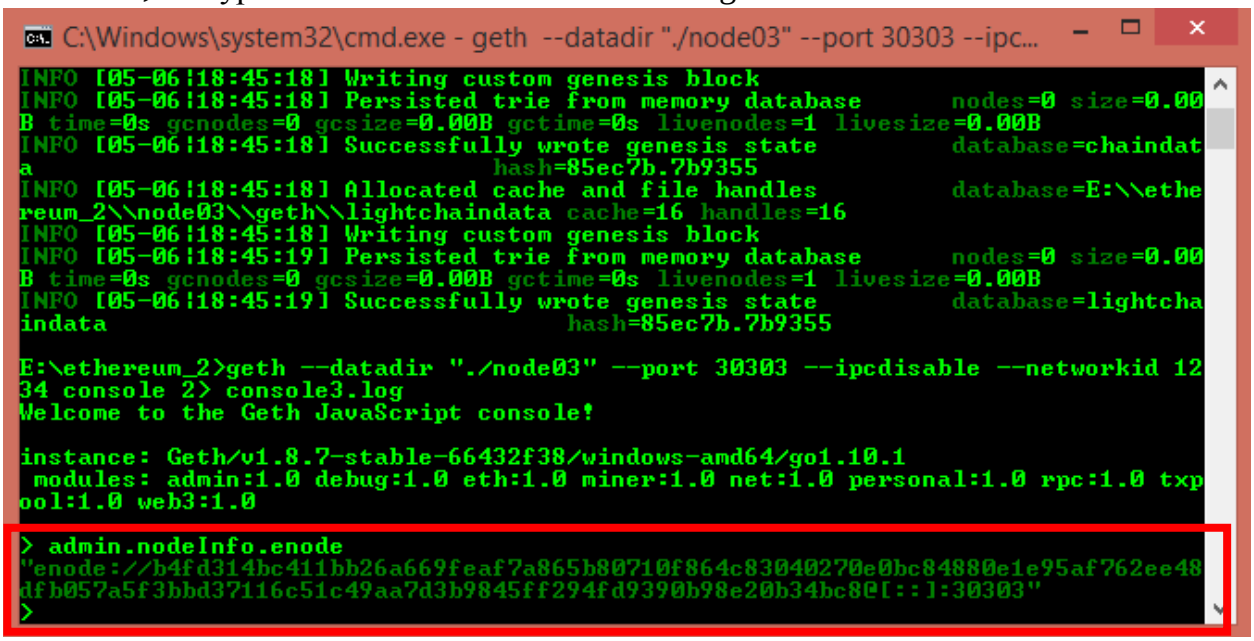

Figure 13. Get enode address at node 2

At **node 1,** we type “admin.addPeer(enode address **node 3**) like Figure 14.

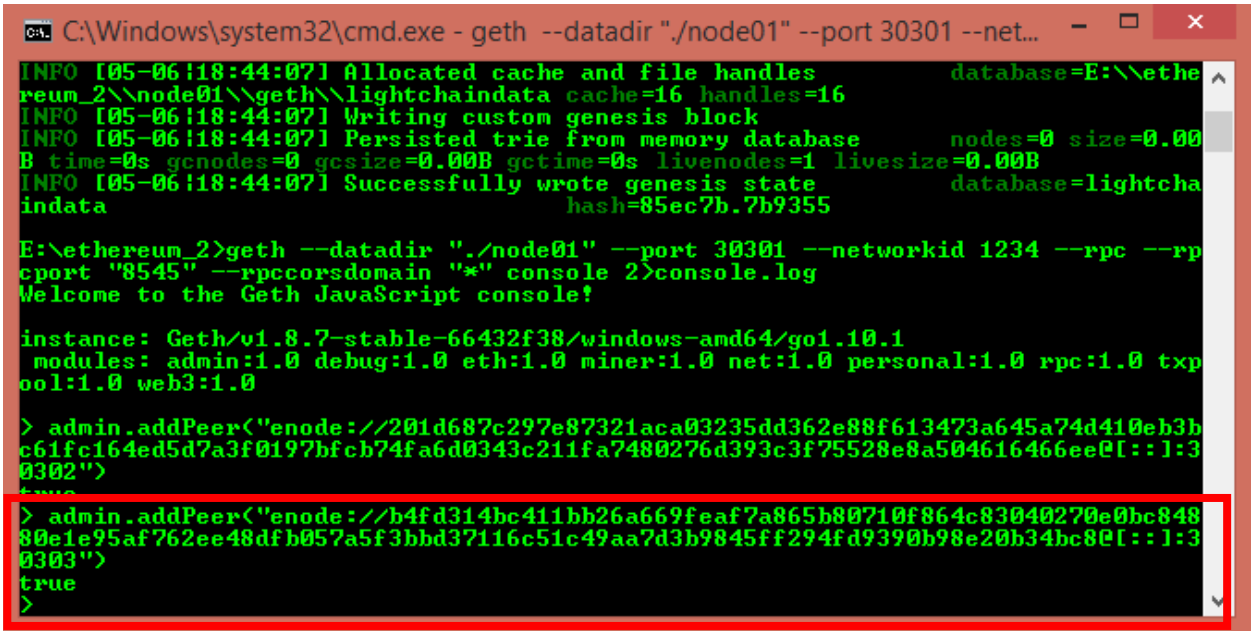

Figure 14. Add node 3 at node 1

Now, you can check that **node 1** sees **node 2** and **node 3** as its **peers** by typing "net.peerCount" and "admin.peers" as shown in Figure 15.

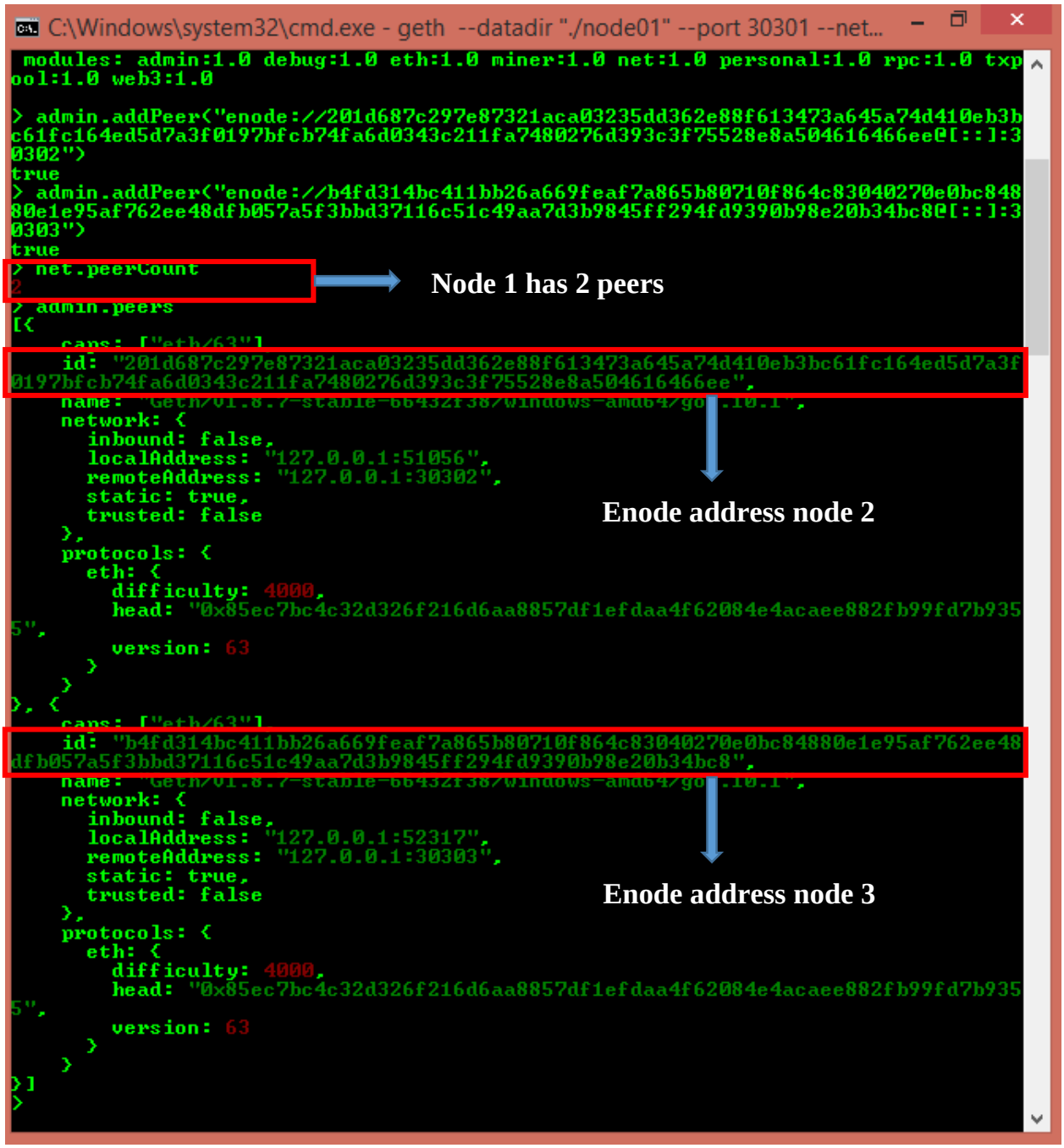

Figure 15. Check peer-to-peer network

Smart contract

Create account

To be able to send transactions as well as mine, we must create an account at each node. We type "*personal.newAccount()*" at each node to create the account like Figure 16.

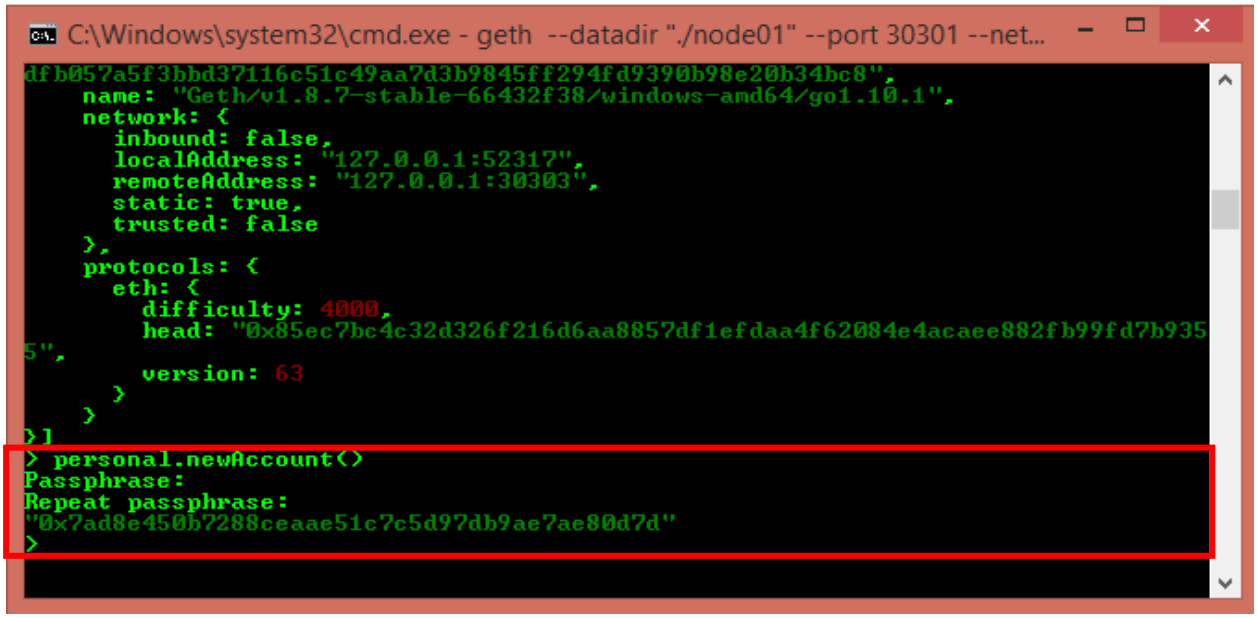

Figure 16. Create account at node 1

Then we have to unlock account to transmit transaction or mine by typing "*personal.unlockAccount('account','')*" as shown in Figure 17, just like login your account into facebook to post status or chat.

Figure 17. Unlock account at node 1

Create smart contract

We open a web browser (Google Chrome,Internet Explorer,...) and type address https://remix.ethereum.org like Figure 18.

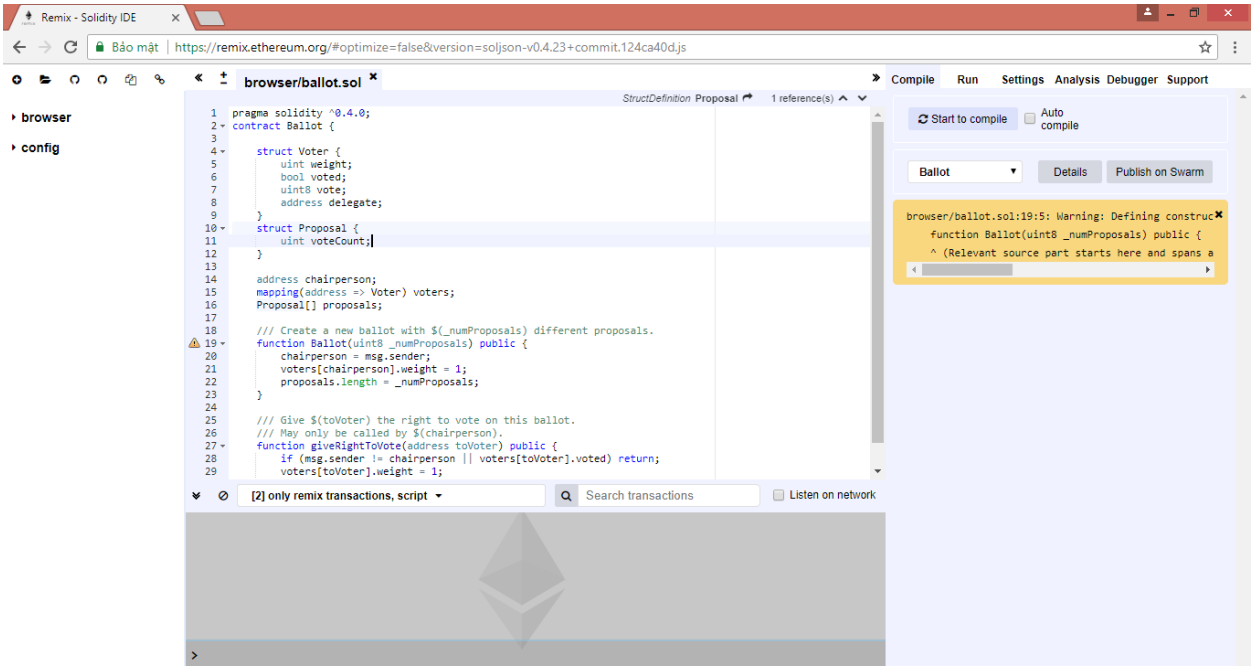

Figure 18. Solidity IDE web

Then we choose “Create New File in the Browser Storage Explore” and rename “mycontract” as shown in Figure 19.

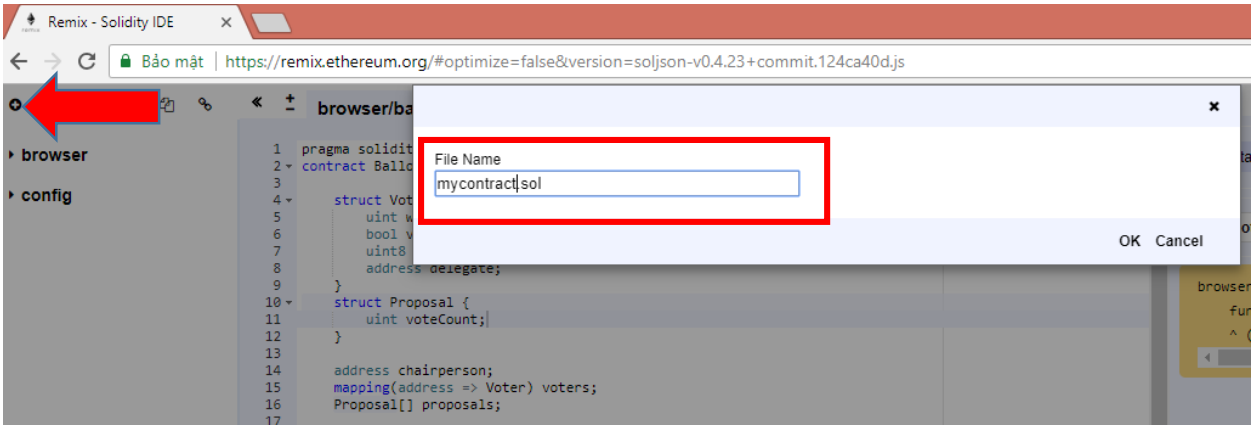

Figure 19. Create a new contract file

Then we write this contract with contents like:

```
pragma solidity ^0.4.16;
contract Money {
   mapping (address => uint) public MoneyBalanceOf;
   event MoneyTransfer(address sender, address receiver, uint amount);

  /* Initializes contract with initial supply money to the creator of the contract */
  function Money(uint supply) {
      MoneyBalanceOf[msg.sender] = supply;
   }

  /* Very simple trade function */
   function sendMoney(address receiver, uint amount) returns(bool sufficient) {
      if (MoneyBalanceOf[msg.sender] < amount) return false;
      MoneyBalanceOf[msg.sender] -= amount;
      MoneyBalanceOf[receiver] += amount;
      MoneyTransfer(msg.sender, receiver, amount);
      return true;
   }
}
```

Then we select "Start to compile" like Figure 20.

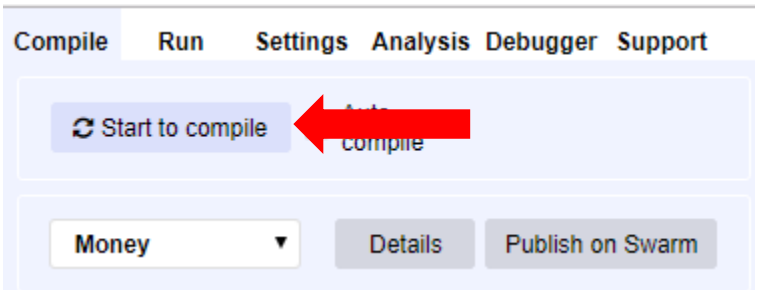

Figure 20. Start to compile

Afterwards, we choose “Run” -> “Enviroment” -> “Web3 Provider” like Figure 21.

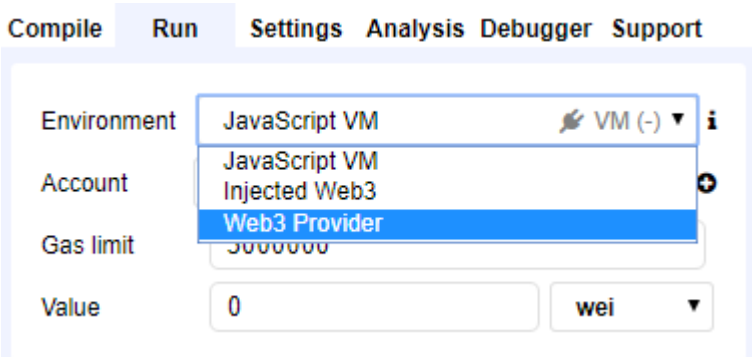

Figure 21s. Choose a Web3 Provider Environment

Then the message will appear like Figure 22, we click OK.

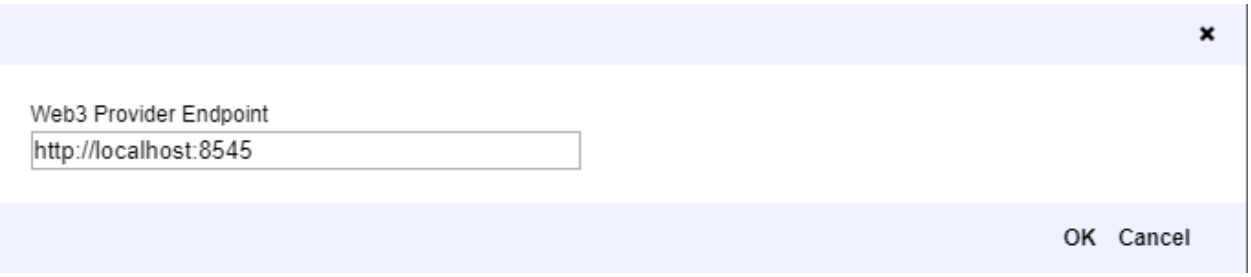

Figure 22. Select local host

Finally, the Web3 Provider will add our Ethereum network with 3 nodes like Figure 23.

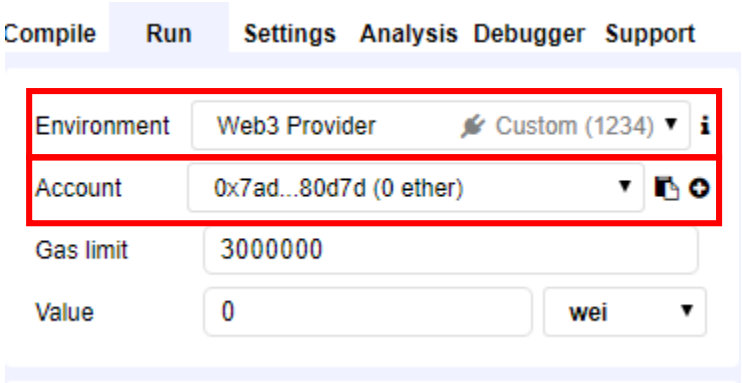

Figure 23. Complete connecting with Ethereum network

Accounts created in the previous steps do not have any ether, meaning that the balance is 0. How do we know that? The account of node 1 have 0 ether in Figure 23. If your account doesn't have any ethers, you cann't totally send the transaction. So I have a little trick to get some ethers as follow: You type "*miner.start (1)"* in few minutes as shown in Figure 24. Then type "*miner.stop()*".

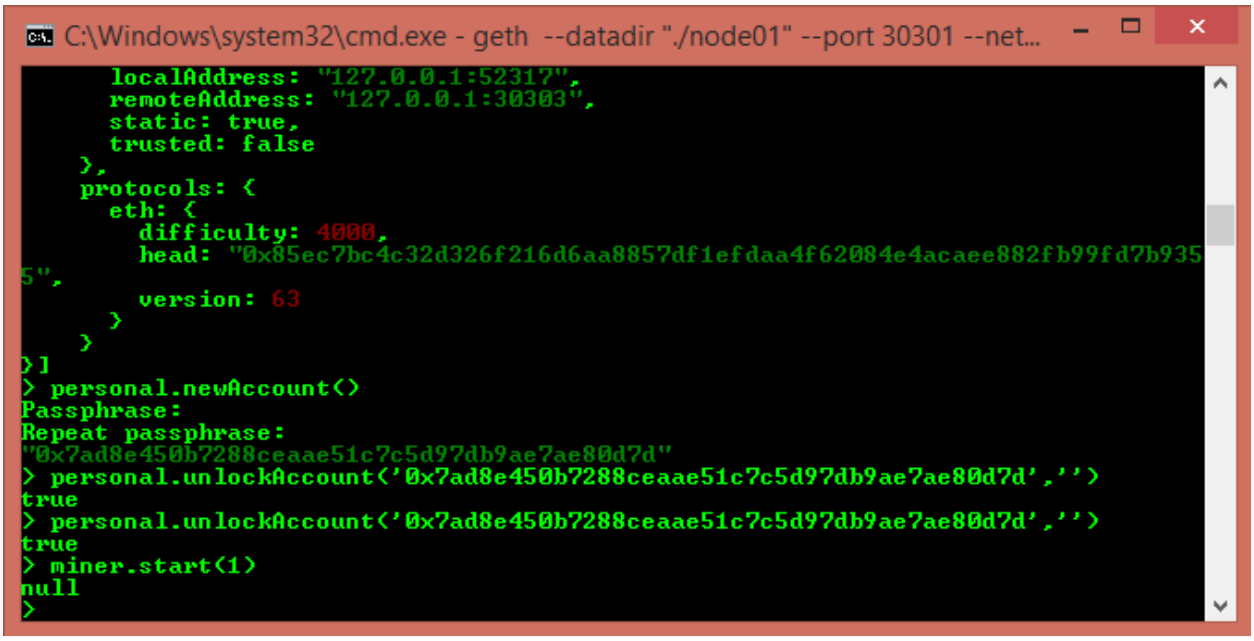

Figure 24. Get ether

Now we can see that the account of node 1 has some ethers like Figure 25.

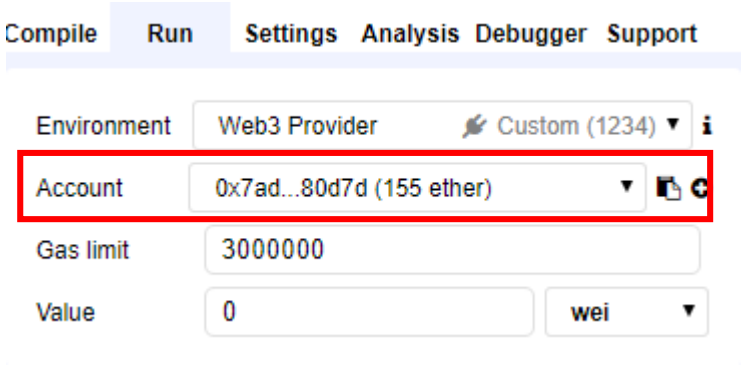

Figure 25. Check ethers at the account of node 1

Now, let's say I have $ 1,000 in the account of node 1 by typing 1000 in this empty boxes as shown in Figure 26. After that we click **Create**.

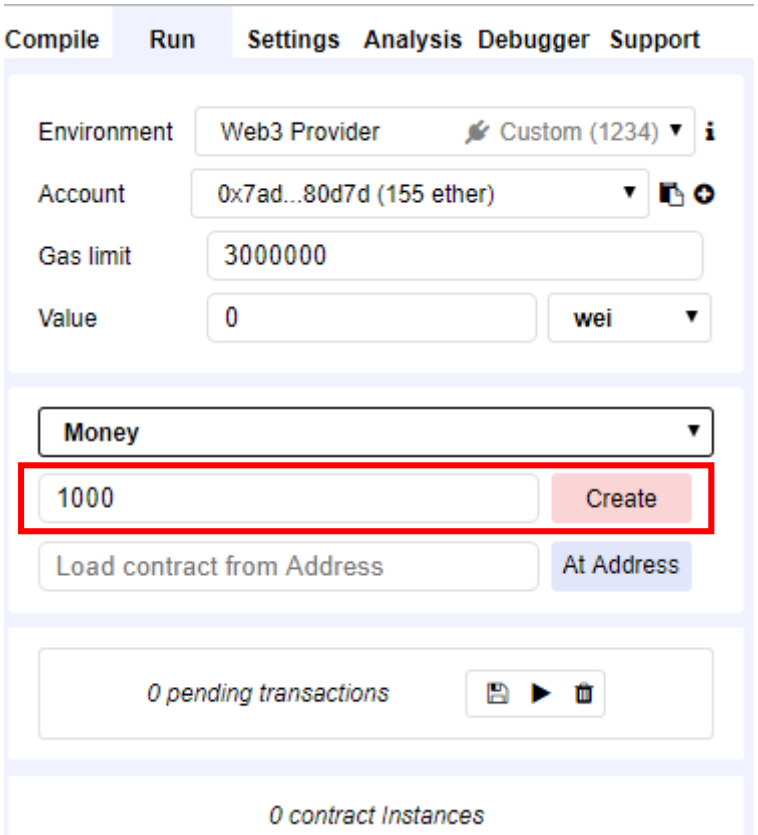

Figure 25. Recharge for the account of node 1

At **node 3**, we type “miner.start(1)” to mine the transaction of recharging for the account of node 1 as shown in Figure 26.

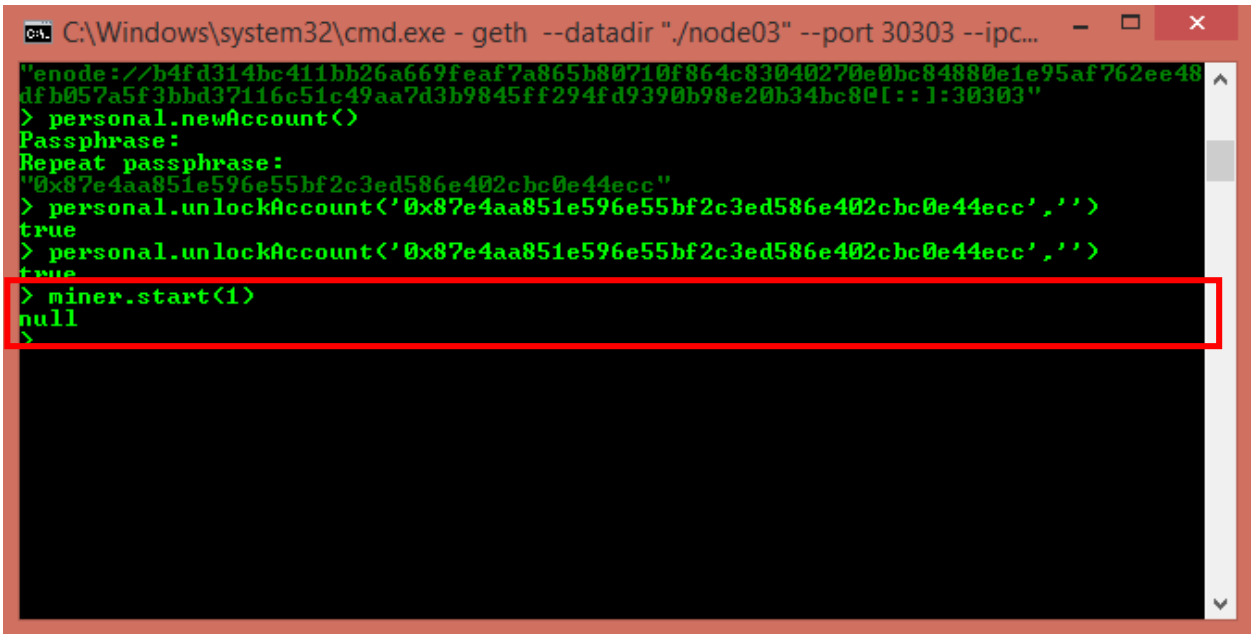

Figure 26. Mine at the account of node 3

In the Solidity IDE browser, we can see the result of miner like Figure 27.

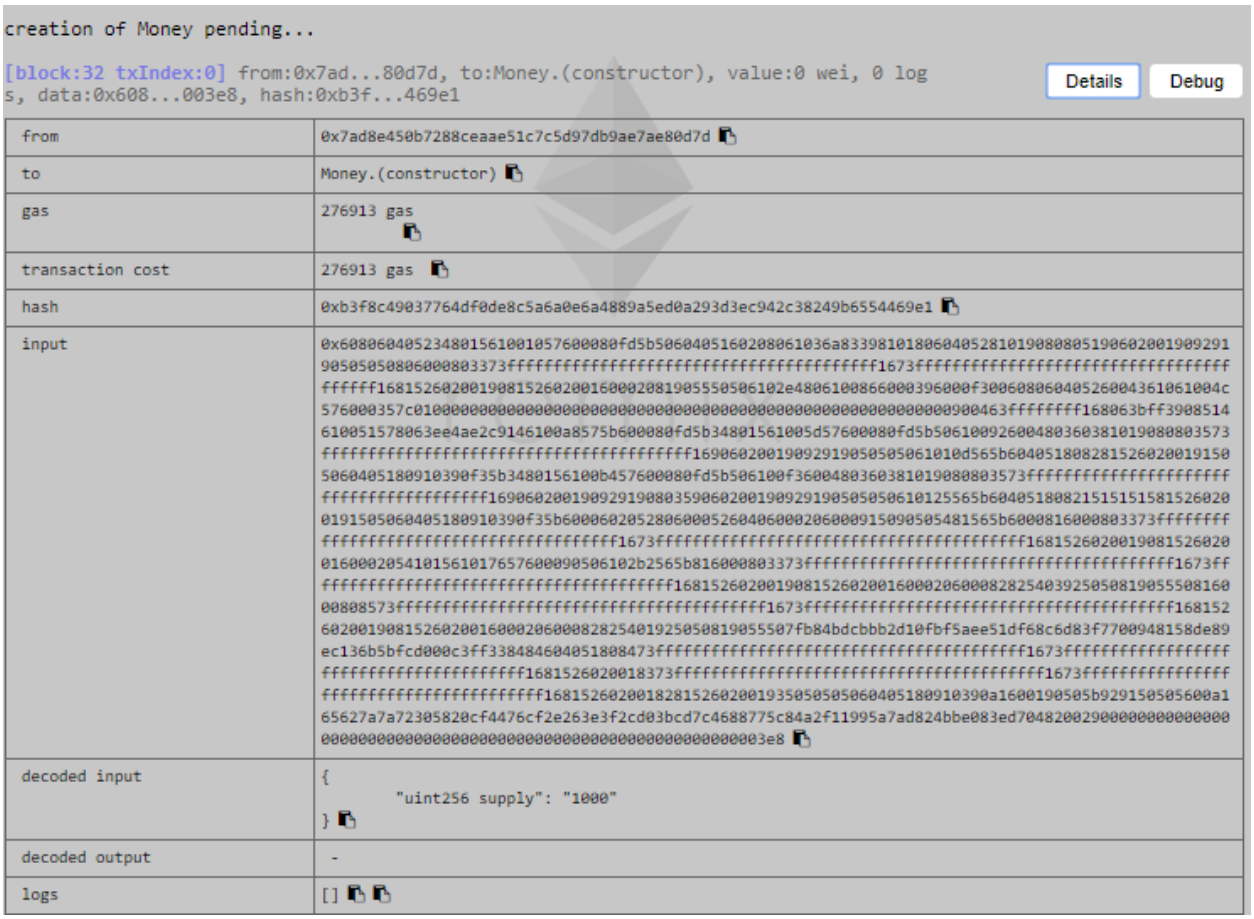

Figure 27. Mining result

After finishing mine of the recharge for the account of node 1, we can check by typing the account of node 1 into “MoneyBalanceOf” box like Figure 28.

Figure 28. Check money at the account of node 1

Now the account of node 1 wants to send money to the account of node 2, but we check firstly the account of node 2 by typing "eth.accounts" like Figure 29.

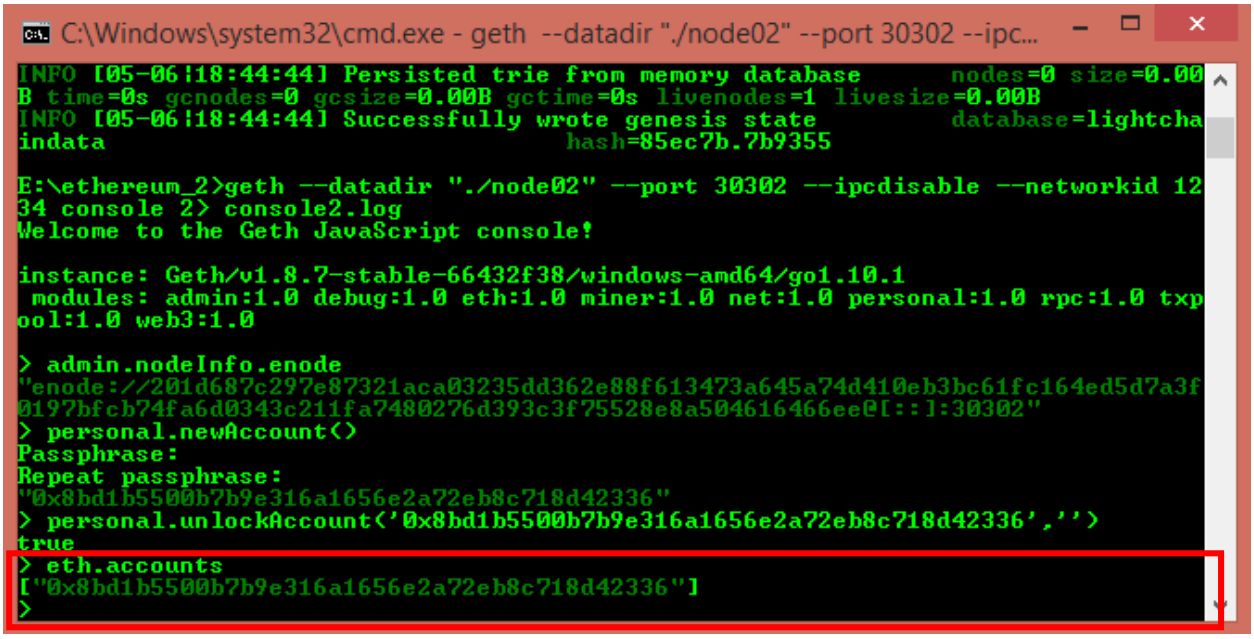

Figure 29. Check the account of node 2

Then the account of node 1 sends $ 400 to the account of node 2 by typing "account of nod 2, 400" into "sendMoney" box like Figure 30.

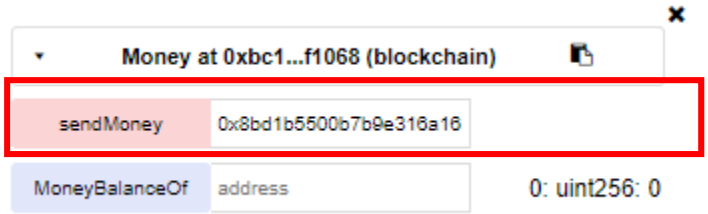

Figure 29. Send Money

At **node 3**, we type "miner.start(1)" to mine the transaction of sending money from the account of node 1 to the account of node 2 like Figure 26.

Now we check the account of node 1 and node 2 as shown in Figure 30.

The account of node 1

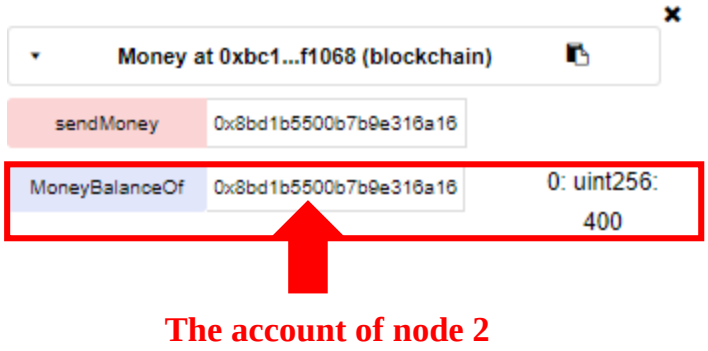

Figure 29. Check Money

Conclusion

In this tutorial, we have totally comprehended insight into Ethereum basic.

Reference

https://ethereum.gitbooks.io/frontier-guide/content/example_script.html