Ethereum Development Tutorial

Written by: Pham Hoai Luan

Last revision: May, 2018

How to create a local private multi-node Ethereum network (geth console)

In the previous tutorial, I have successfully completed the installation of Geth on Windows. To continue our work, let's go deeper into the Ethereum.

Now we can access the Ethereum protocol via the nodes we create ourselves. Afterthat, we can create a peer-to-peer network to develop and deploy smart contracts as shown in Figure 1. Note that we are building on the fake Ethereum protocol, not the real Ethereum.

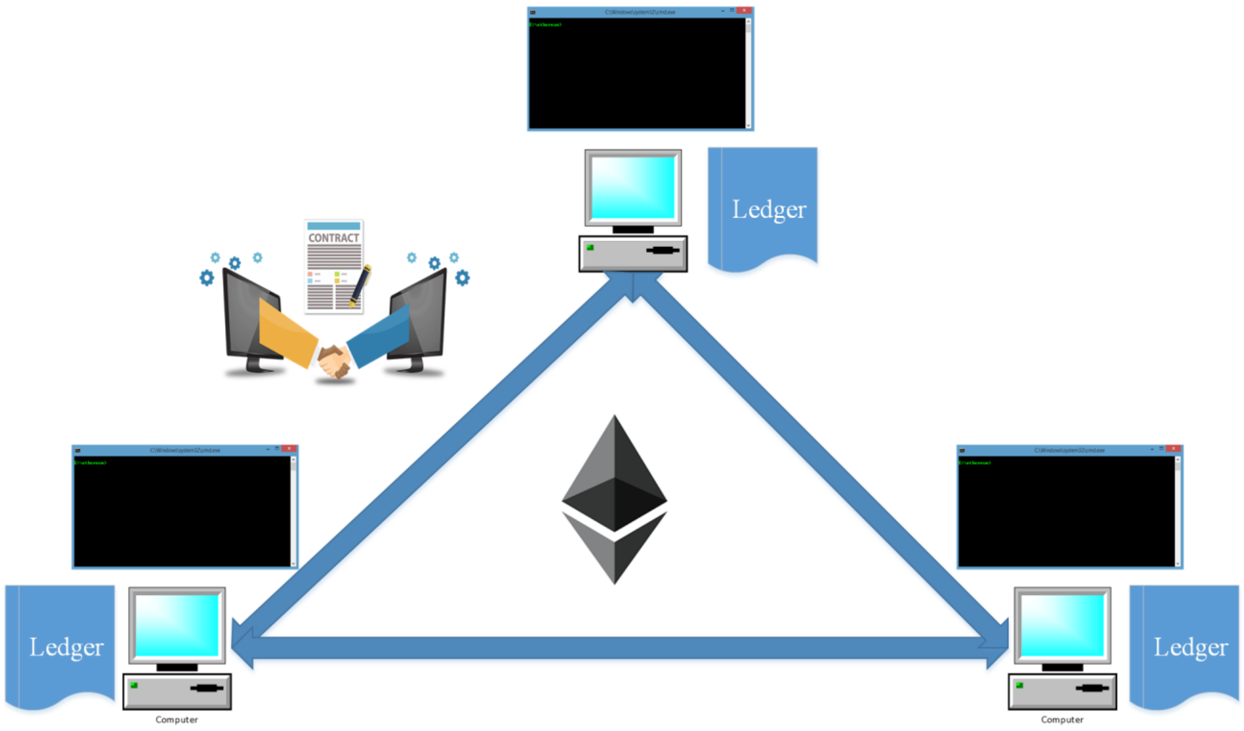


Figure 1. Basic ethereum model

Create genesis.json file

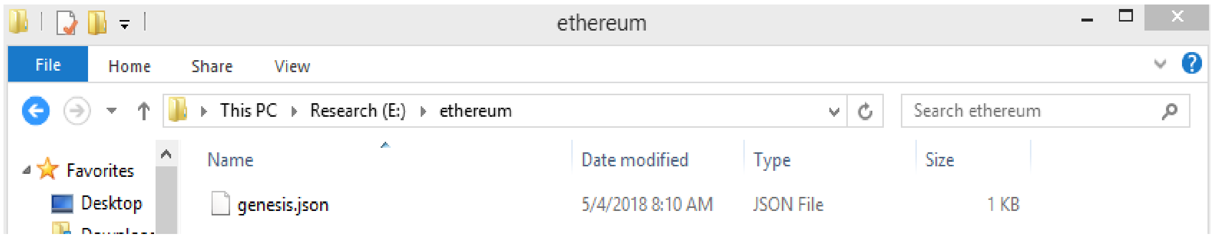
Every blockchain starts with a Genesis Block, the very first block in the chain, block ZERO— the only block that does not have a predecessor.

To create our private blockchain then, we will create a genesis block. To do this, we will create a custom genesis file, and ask Geth to use that genesis file to create our own genesis block, which in turn will be the start of our custom private blockchain.

Here’s what a genesis file [1] looks like:

```
{
"config":{
        "chainId": 2018,
        "homesteadBlock": 0,
        "eip155Block": 0,
        "eip158Block": 0,
        "byzantiumBlock": 12
    },
  "alloc"      : {},
  "coinbase"   : "0x0000000000000000000000000000000000000000",
  "difficulty" : "0x20000",
  "extraData"  : "",
  "gasLimit"   : "0x2fefd8",
  "nonce"      : "0x0000000000000042",
  "mixhash"    :
"0x0000000000000000000000000000000000000000000000000000000000000000",
  "parentHash" :
"0x0000000000000000000000000000000000000000000000000000000000000000",
  "timestamp"  : "0x00"
}
```

Now we create a folder "ethereum" to work as shown in Figure 2 and save the generated genesis.json file.

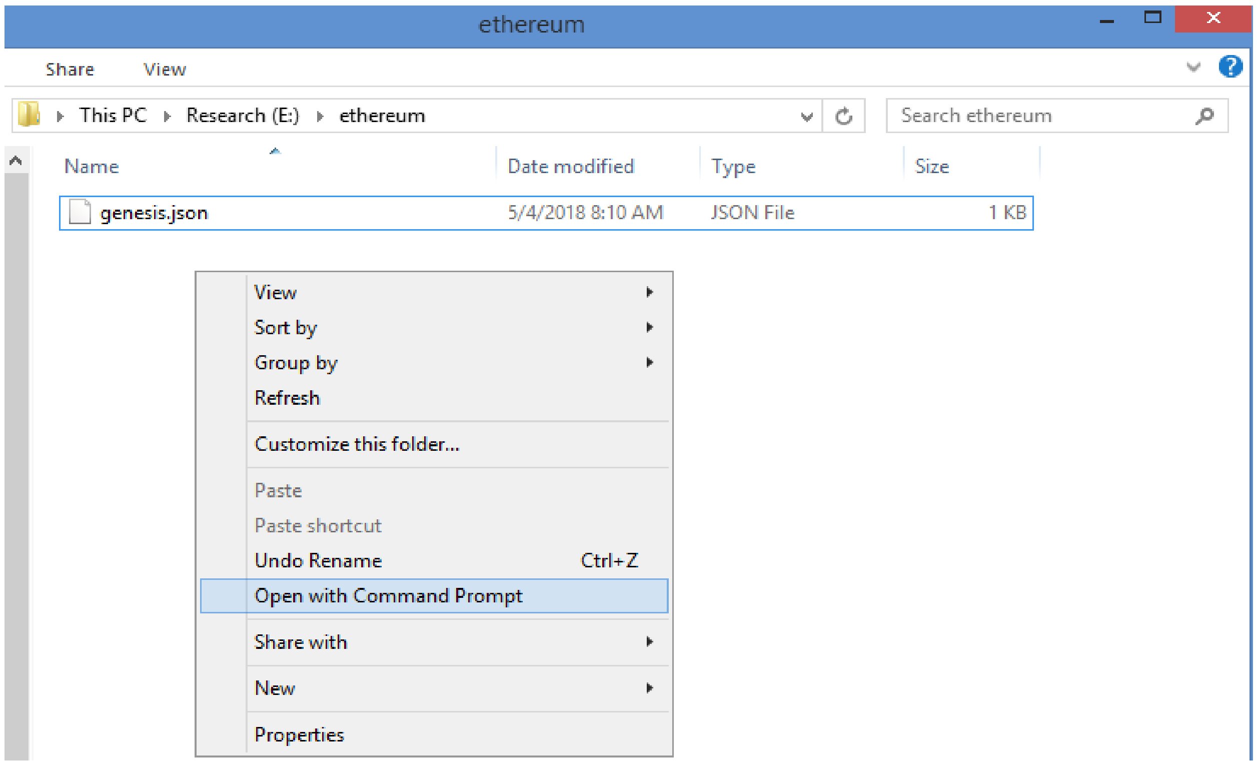


Name	Date modified	Type	Size
genesis.json	5/4/2018 8:10 AM	JSON File	1 KB

Figure 2. Create working directory

Init GETH to create the genesis block [2]

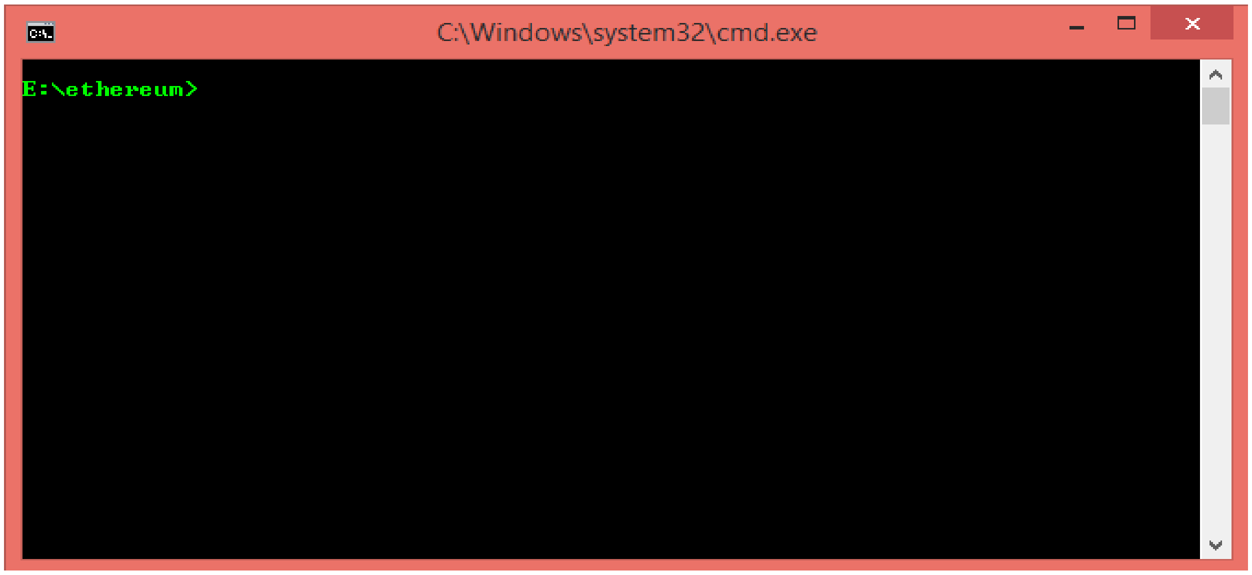
Before creating the private node, you have to create the node containing the genesis block. We go to the "ethereum" folder, right-click and choose Open with Command Prompt as shown in Figure 3.



Name	Date modified	Type	Size
genesis.json	5/4/2018 8:10 AM	JSON File	1 KB

Figure 3. Open Command Prompt

Then, Command Prompt terminal will appear as Figure 4.



```
E:\ethereum>
```

Figure 4. Command Prompt terminal

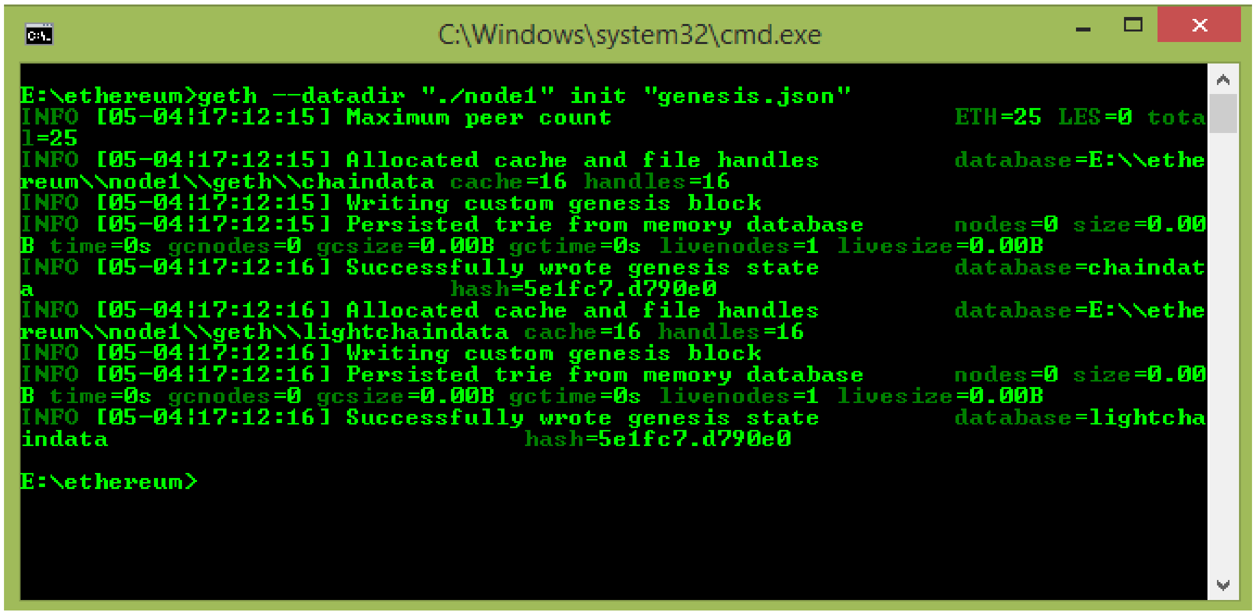
Each terminal is our node (computer, laptop, etc), now we suppose that we need to create two nodes. Therefore, we will open a new Command Prompt terminal. Then, we will get two nodes as shown in Figure 5.

Node 1
Node 2
C:\Windows\system32\cmd.exe
E:\ethereum>
C:\Windows\system32\cmd.exe
E:\ethereum>
Computer
Computer

Figure 5. Two Command Prompt terminals

Starting node 1

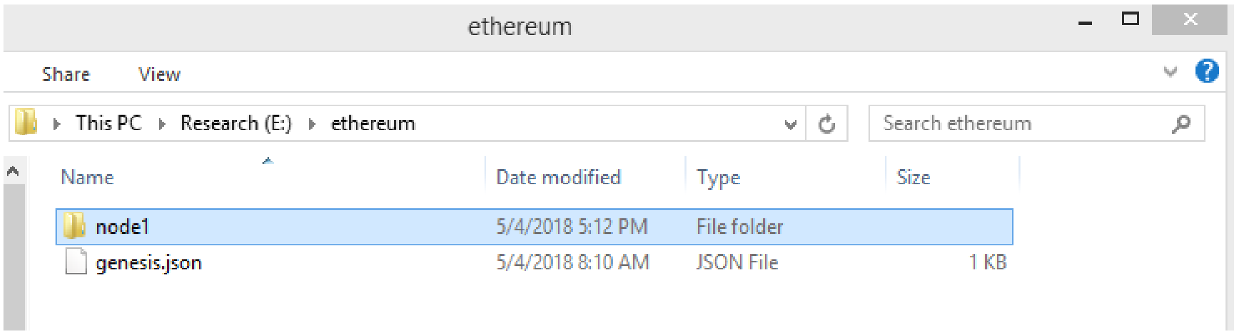
Type " *geth --datadir "./node1" init "genesis.json"* " into the Command Prompt terminal of node 1 as shown in Figure 5.



```
E:\ethereum>geth --datadir "./node1" init "genesis.json"
INFO [05-04|17:12:15] Maximum peer count                       ETH=25 LES=0 tota
l=25
INFO [05-04|17:12:15] Allocated cache and file handles         database=E:\\ethe
reum\\node1\\geth\\chaindata cache=16 handles=16
INFO [05-04|17:12:15] Writing custom genesis block
INFO [05-04|17:12:15] 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 [05-04|17:12:16] Successfully wrote genesis state         database=chaindat
a                            hash=5e1fc7.d790e0
INFO [05-04|17:12:16] Allocated cache and file handles         database=E:\\ethe
reum\\node1\\geth\\lightchaindata cache=16 handles=16
INFO [05-04|17:12:16] Writing custom genesis block
INFO [05-04|17:12:16] 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 [05-04|17:12:16] Successfully wrote genesis state         database=lightcha
indata                                 hash=5e1fc7.d790e0

E:\ethereum>
```

Figure 5. Geth init at node 1

Then the command will create automatically a "node1" folder in the "ethereum" directory as shown in Figure 6.

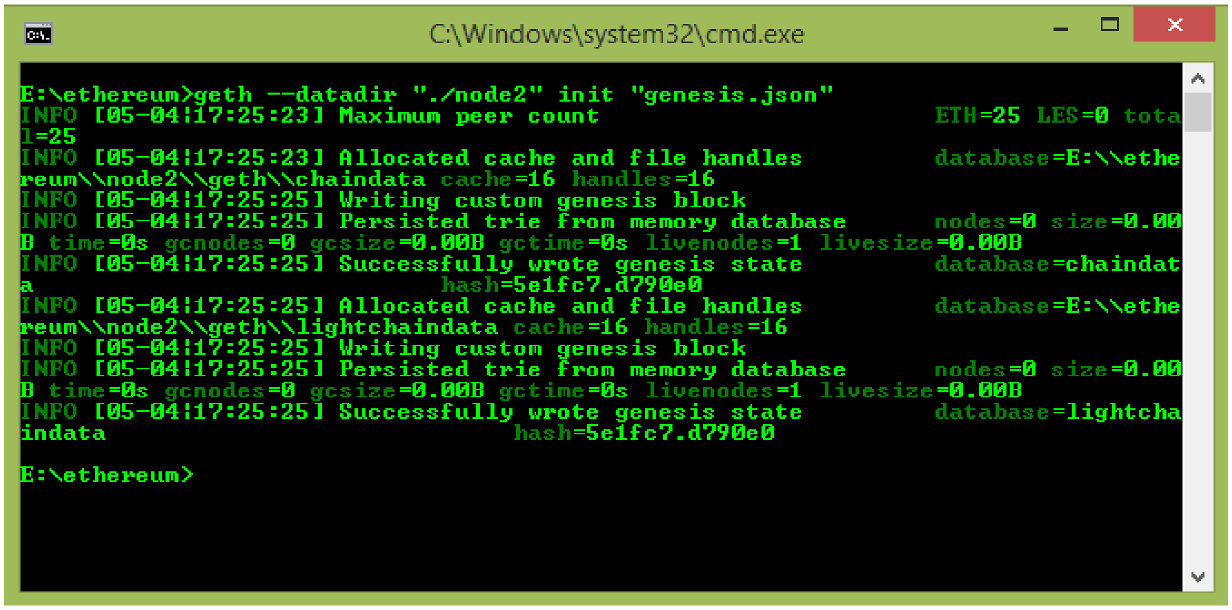


Name	Date modified	Type	Size
node1	5/4/2018 5:12 PM	File folder	
genesis.json	5/4/2018 8:10 AM	JSON File	1 KB

Figure 6. A “node1” folder is created after making command

Starting node 2

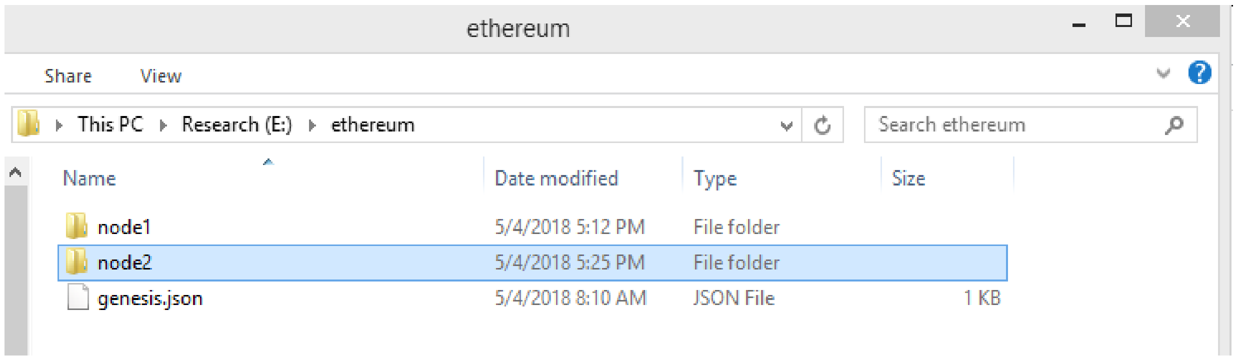
Type " *geth --datadir "./node2" init "genesis.json"* " into the Command Prompt terminal of node 1 as shown in Figure 7.



```
E:\ethereum>geth --datadir "./node2" init "genesis.json"
INFO [05-04|17:25:23] Maximum peer count                       ETH=25 LES=0 tota
l=25
INFO [05-04|17:25:23] Allocated cache and file handles         database=E:\\ethe
reum\\node2\\geth\\chaindata cache=16 handles=16
INFO [05-04|17:25:25] Writing custom genesis block
INFO [05-04|17:25:25] 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 [05-04|17:25:25] Successfully wrote genesis state         database=chaindat
a                            hash=5e1fc7.d790e0
INFO [05-04|17:25:25] Allocated cache and file handles         database=E:\\ethe
reum\\node2\\geth\\lightchaindata cache=16 handles=16
INFO [05-04|17:25:25] Writing custom genesis block
INFO [05-04|17:25:25] 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 [05-04|17:25:25] Successfully wrote genesis state         database=lightcha
indata                            hash=5e1fc7.d790e0

E:\ethereum>
```

Figure 7. Geth init at node 2

Then the command will create automatically a "node2" folder in the "ethereum" directory as shown in Figure 8.

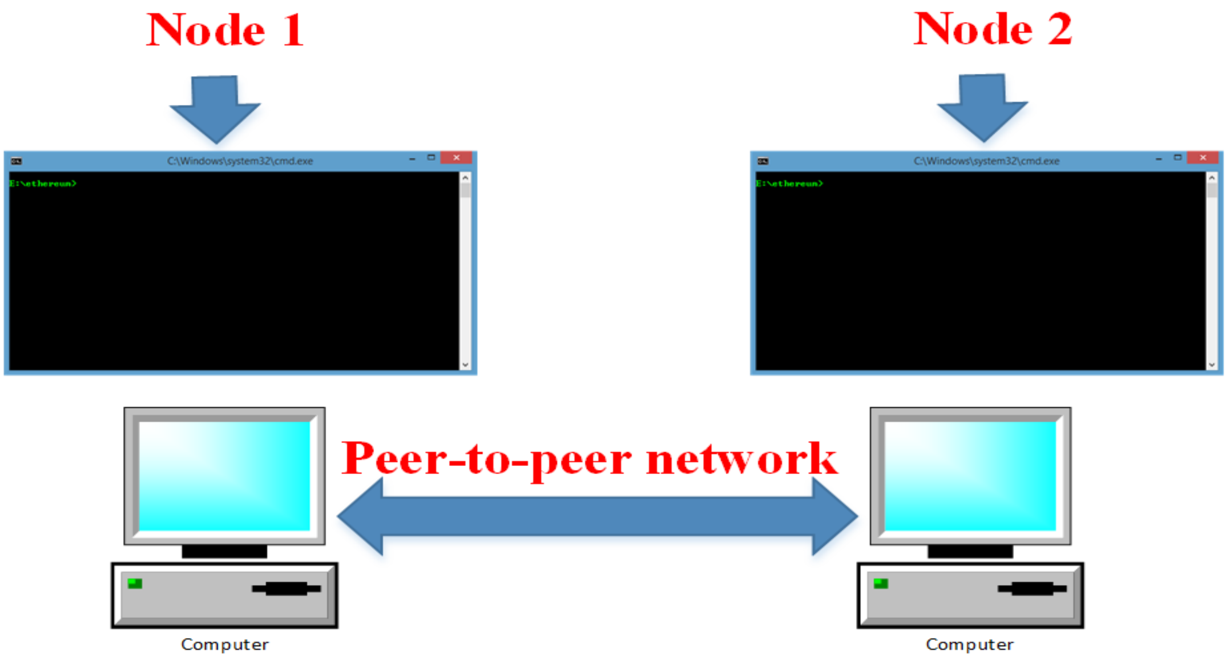


Name	Date modified	Type	Size
node1	5/4/2018 5:12 PM	File folder	
node2	5/4/2018 5:25 PM	File folder	
genesis.json	5/4/2018 8:10 AM	JSON File	1 KB

Figure 8. A "node2" folder is created after making command

Create a private network with 2 nodes [2]

After completing the genesis block in each node, they are connected together to form a simple peer-to-peer network as shown in Figure 9.



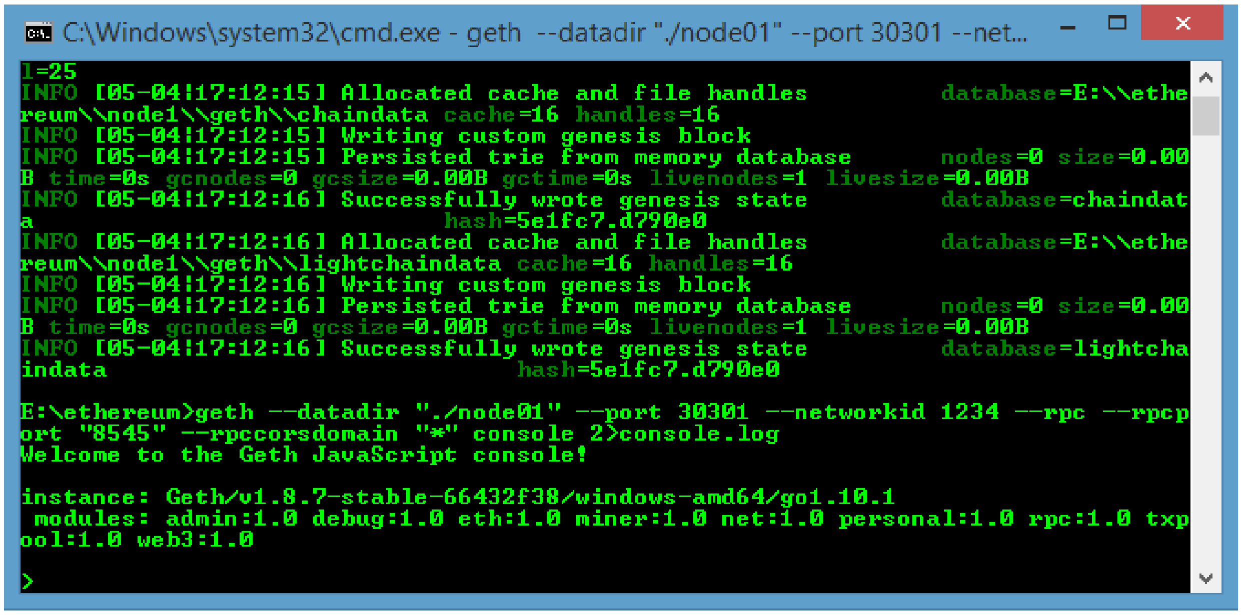
```
E:\ethereum>
```

Figure 9. Peer-to-peer network

To make the peer-to-peer network, we follow these steps:

Starting node 1

Type " *geth --datadir "./node1" --port 30301 --networkid 1234 --rpc --rpcport "8545" --rpccorsdomain "*" console 2>console.log* " into the Command Prompt terminal of node 1 as shown in Figure 10.

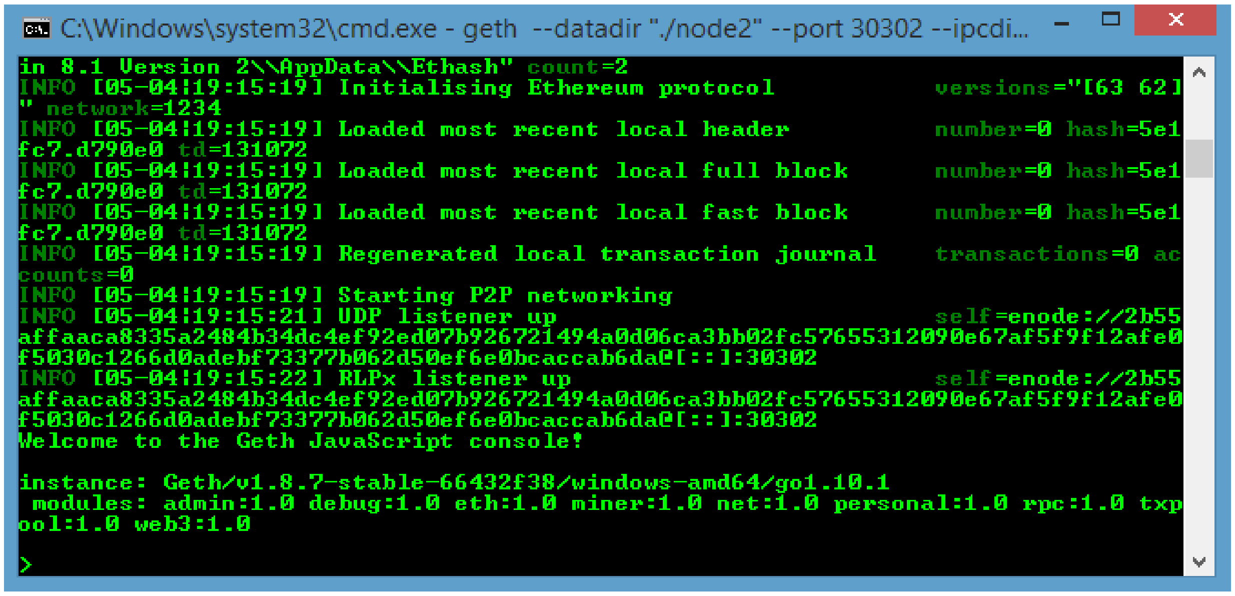


```
C:\Windows\system32\cmd.exe - geth  --datadir "./node01" --port 30301 --net...
l=25
INFO [05-04|17:12:15] Allocated cache and file handles         database=E:\\ethe
reum\\node1\\geth\\chaindata cache=16 handles=16
INFO [05-04|17:12:15] Writing custom genesis block
INFO [05-04|17:12:15] Persisted 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 [05-04|17:12:16] Successfully wrote genesis state         database=chaindat
a                                                        hash=5e1fc7.d790e0
INFO [05-04|17:12:16] Allocated cache and file handles         database=E:\\ethe
reum\\node1\\geth\\lightchaindata cache=16 handles=16
INFO [05-04|17:12:16] Writing custom genesis block
INFO [05-04|17:12:16] 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 [05-04|17:12:16] Successfully wrote genesis state         database=lightcha
indata                                                        hash=5e1fc7.d790e0

E:\ethereum>geth --datadir "./node01" --port 30301 --networkid 1234 --rpc --rpcp
ort "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

>
```

Figure 10. Geth console at node 1

Starting node 2

Type " *geth --datadir "./node2" --port 30302 --ipcdisable --networkid 1234 console*" into the Command Prompt terminal of node 2 as shown in Figure 11.

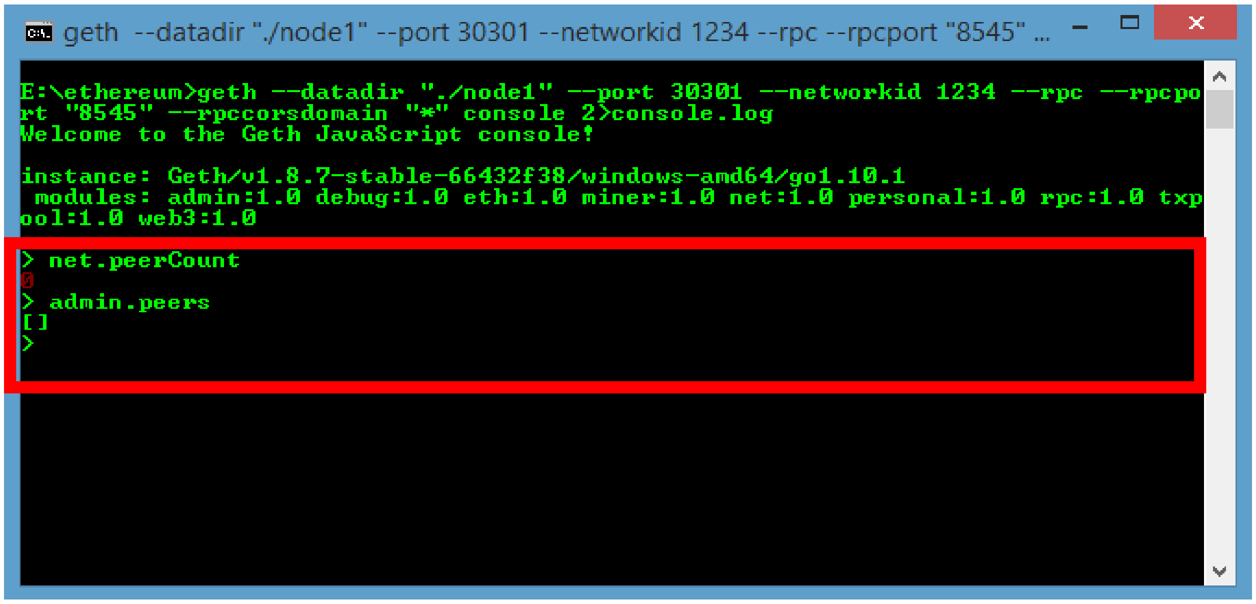


```
C:\Windows\system32\cmd.exe - geth  --datadir "./node2" --port 30302 --ipcdi...
in 8.1 Version 2\\AppData\\Ethash" count=2
INFO [05-04|19:15:19] Initialising Ethereum protocol           versions="[63 62]
" network=1234
INFO [05-04|19:15:19] Loaded most recent local header          number=0 hash=5e1
fc7.d790e0 td=131072
INFO [05-04|19:15:19] Loaded most recent local full block      number=0 hash=5e1
fc7.d790e0 td=131072
INFO [05-04|19:15:19] Loaded most recent local fast block      number=0 hash=5e1
fc7.d790e0 td=131072
INFO [05-04|19:15:19] Regenerated local transaction journal    transactions=0 ac
counts=0
INFO [05-04|19:15:19] Starting P2P networking
INFO [05-04|19:15:21] UDP listener up                          self=enode://2b55
affaaca8335a2484b34dc4ef92ed07b926721494a0d06ca3bb02fc57655312090e67af5f9f12afe0
f5030c1266d0adebf73377b062d50ef6e0bcaccab6da@[::]:30302
INFO [05-04|19:15:22] RLPx listener up                         self=enode://2b55
affaaca8335a2484b34dc4ef92ed07b926721494a0d06ca3bb02fc57655312090e67af5f9f12afe0
f5030c1266d0adebf73377b062d50ef6e0bcaccab6da@[::]:30302
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

>
```

Figure 11. Geth console at node 2

Linking the two Geth nodes

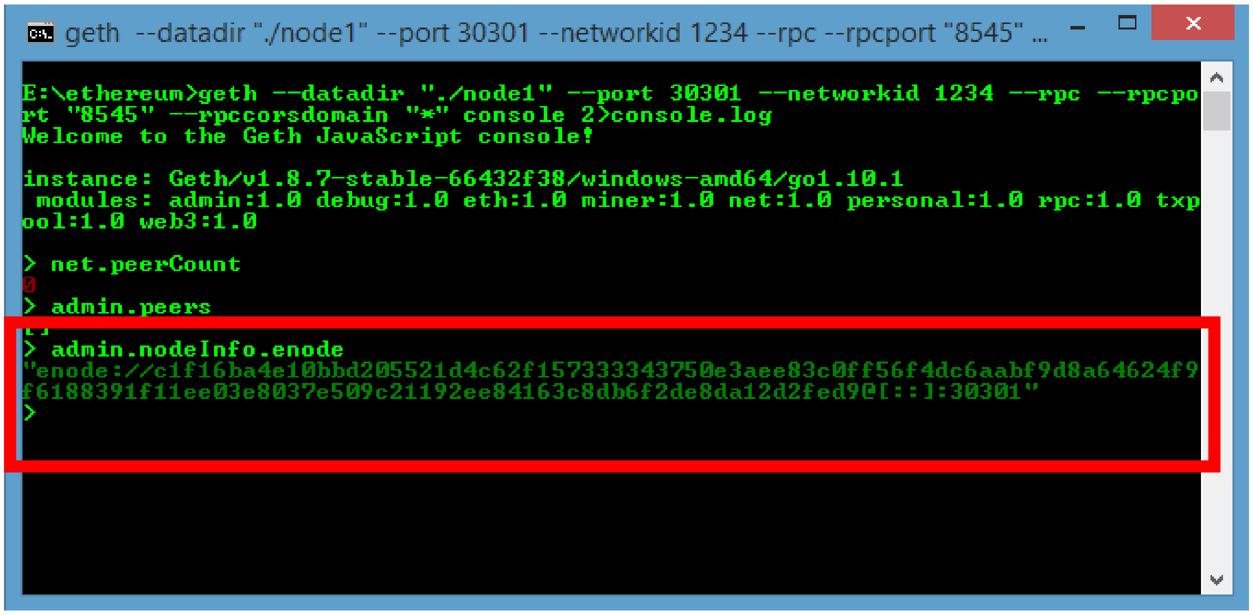
Two nodes aren't currently connected to each other. At the node 1, we probaly type the following command "net.peerCount" to see how many nodes you have in your network or "admin.peers" to show all nodes at the network. The result,as shown in Figure 12, indicates that two nodes are completely unconnected.



```
geth  --datadir "./node1" --port 30301 --networkid 1234 --rpc --rpcport "8545" ...

E:\ethereum>geth --datadir "./node1" --port 30301 --networkid 1234 --rpc --rpcpo
rt "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

> net.peerCount
0
> admin.peers
[]
>
```

Figure 12. Check network connection at node 1

At node 1, at the command window console we type "*admin.nodeInfo.enode*" to show the enode address as shown in Figure 13.



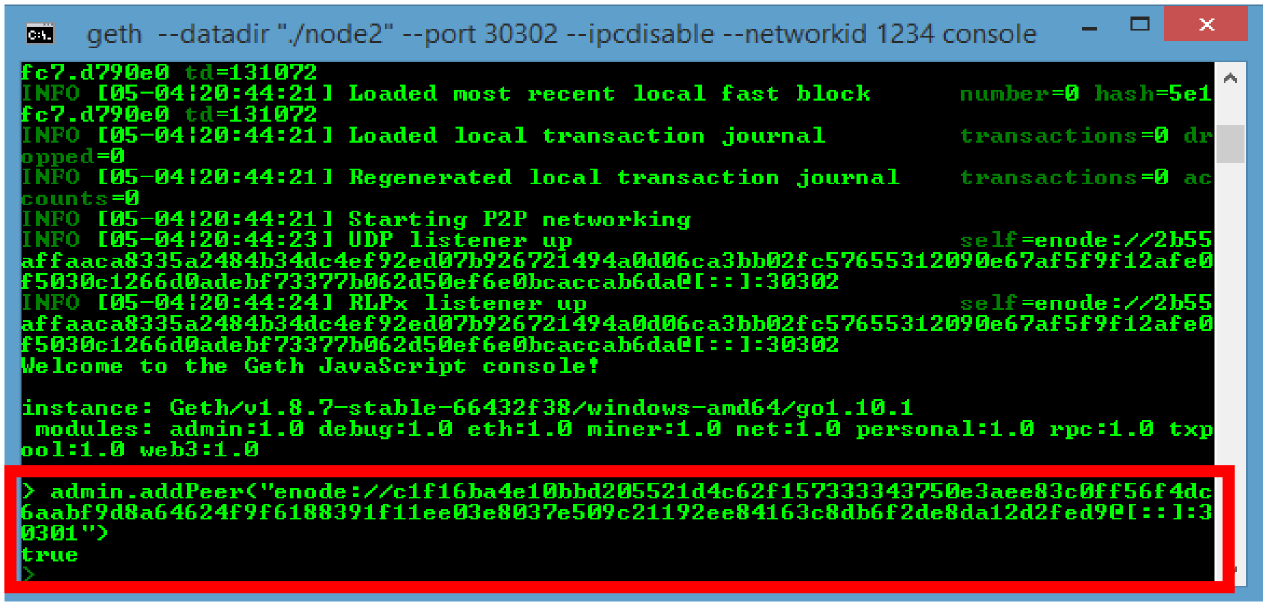
```
geth  --datadir "./node1" --port 30301 --networkid 1234 --rpc --rpcport "8545" ...

E:\ethereum>geth --datadir "./node1" --port 30301 --networkid 1234 --rpc --rpcpo
rt "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

> net.peerCount
0
> admin.peers
[]
> admin.nodeInfo.enode
"enode://c1f16ba4e10bbd205521d4c62f157333343750e3aee83c0ff56f4dc6aabf9d8a64624f9
f6188391f11ee03e8037e509c21192ee84163c8db6f2de8da12d2fed9@[::]:30301"
>
```

Figure 13. Enode address at node 1

To connect two nodes together, we copy enode address of node 1 as shown in Figure 13 and then type admin.addPeer ("enode address of node 1") at node 2 console as shown in Figure 14.

For example:

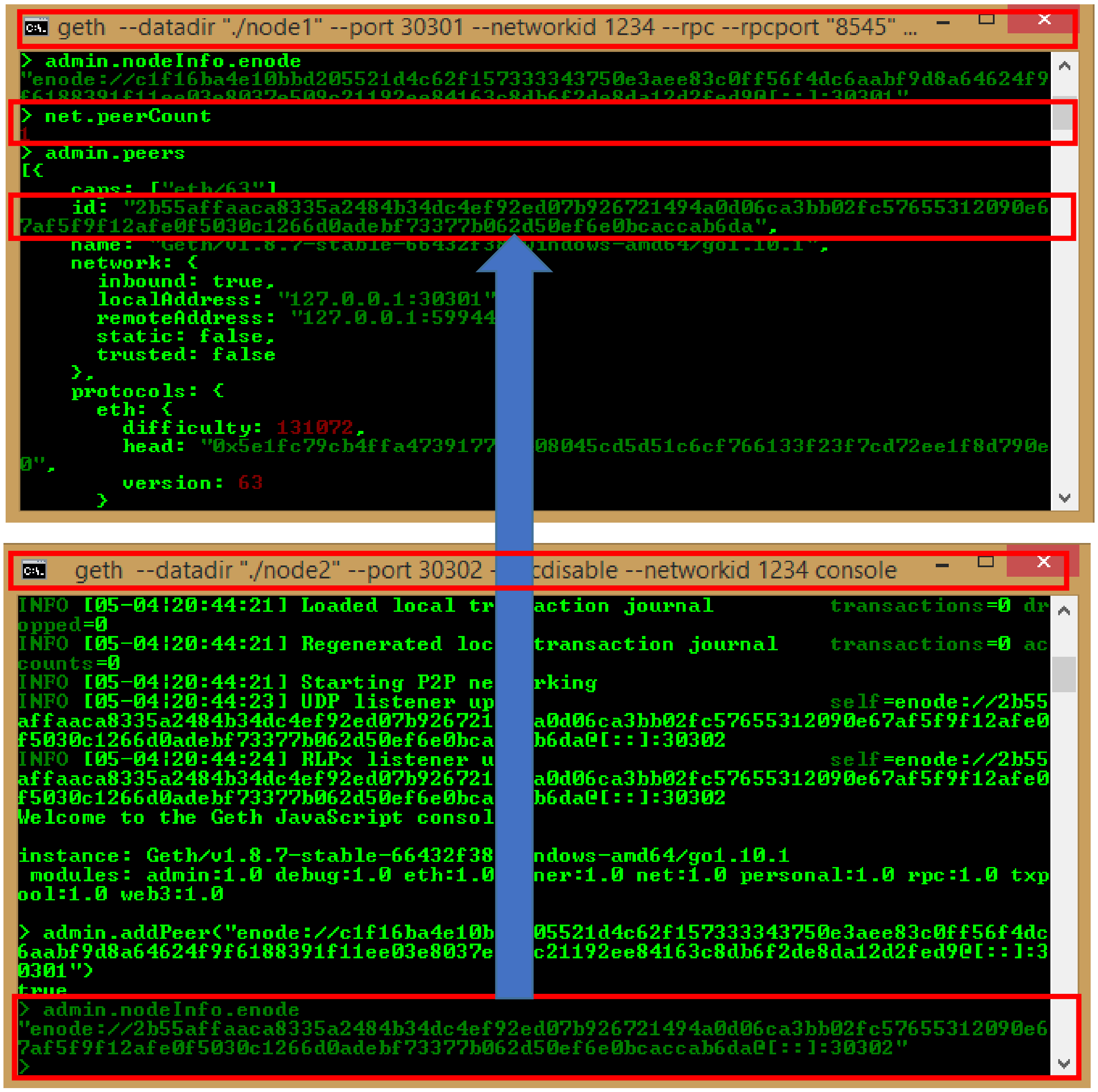
admin.addPeer("enode://c1f16ba4e10bbd205521d4c62f157333343750e3aee83c0ff56f4d c6aabf9d8a64624f9f6188391f11ee03e8037e509c21192ee84163c8db6f2de8da12d2fed9 @[::]:30301")



```
geth  --datadir "./node2" --port 30302 --ipcdisable --networkid 1234 console
fc7.d790e0 td=131072
INFO [05-04|20:44:21] Loaded most recent local fast block      number=0 hash=5e1
fc7.d790e0 td=131072
INFO [05-04|20:44:21] Loaded local transaction journal         transactions=0 dr
opped=0
INFO [05-04|20:44:21] Regenerated local transaction journal    transactions=0 ac
counts=0
INFO [05-04|20:44:21] Starting P2P networking
INFO [05-04|20:44:23] UDP listener up                          self=enode://2b55
affaaca8335a2484b34dc4ef92ed07b926721494a0d06ca3bb02fc57655312090e67af5f9f12afe0
f5030c1266d0adebf73377b062d50ef6e0bcaccab6da@[::]:30302
INFO [05-04|20:44:24] RLPx listener up                         self=enode://2b55
affaaca8335a2484b34dc4ef92ed07b926721494a0d06ca3bb02fc57655312090e67af5f9f12afe0
f5030c1266d0adebf73377b062d50ef6e0bcaccab6da@[::]:30302
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://c1f16ba4e10bbd205521d4c62f157333343750e3aee83c0ff56f4dc
6aabf9d8a64624f9f6188391f11ee03e8037e509c21192ee84163c8db6f2de8da12d2fed9@[::]:3
0301")
true
>
```

Figure 14. Add peer

Finally, we check whether two nodes are connected together by typing "net.peerCount" and "admin.peers" at node 1. In Figure 15, we can see that node 1 has identified a peer-to-peer network with one additional node and the enode address of the additional node is identical to node 2. Thus, we can confirm that node 1 and node 2 have become peer-to-peer network.



```
geth  --datadir "./node1" --port 30301 --networkid 1234 --rpc --rpcport "8545" ...
> admin.nodeInfo.enode
"enode://c1f16ba4e10bbd205521d4c62f157333343750e3aee83c0ff56f4dc6aabf9d8a64624f9
f6188391f11ee03e8037e509c21192ee84163c8db6f2de8da12d2fed90[::]:30301"
> net.peerCount
1
> admin.peers
[{
    caps: ["eth/63"]
    id: "2b55affaaca8335a2484b34dc4ef92ed07b926721494a0d06ca3bb02fc57655312090e6
7af5f9f12afe0f5030c1266d0adebf73377b062d50ef6e0bcaccab6da",
    name: "Geth/v1.8.7-stable-66432f38 windows-amd64/go1.10.1",
    network: {
      inbound: true,
      localAddress: "127.0.0.1:30301"
      remoteAddress: "127.0.0.1:59944
      static: false,
      trusted: false
    },
    protocols: {
      eth: {
        difficulty: 131072,
        head: "0x5e1fc79cb4ffa4739177    08045cd5d51c6cf766133f23f7cd72ee1f8d790e
0",
        version: 63
      }
```

```
geth  --datadir "./node2" --port 30302 -    cdisable --networkid 1234 console
INFO [05-04|20:44:21] Loaded local tr    action journal          transactions=0 dr
opped=0
INFO [05-04|20:44:21] Regenerated loc    transaction journal     transactions=0 ac
counts=0
INFO [05-04|20:44:21] Starting P2P ne    rking
INFO [05-04|20:44:23] UDP listener up                            self=enode://2b55
affaaca8335a2484b34dc4ef92ed07b926721    a0d06ca3bb02fc57655312090e67af5f9f12afe0
f5030c1266d0adebf73377b062d50ef6e0bca    b6da@[::]:30302
INFO [05-04|20:44:24] RLPx listener u                            self=enode://2b55
affaaca8335a2484b34dc4ef92ed07b926721    a0d06ca3bb02fc57655312090e67af5f9f12afe0
f5030c1266d0adebf73377b062d50ef6e0bca    b6da@[::]:30302
Welcome to the Geth JavaScript consol

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

> admin.addPeer("enode://c1f16ba4e10b    05521d4c62f157333343750e3aee83c0ff56f4dc
6aabf9d8a64624f9f6188391f11ee03e8037e    c21192ee84163c8db6f2de8da12d2fed9@[::]:3
0301")
true
> admin.nodeInfo.enode
"enode://2b55affaaca8335a2484b34dc4ef92ed07b926721494a0d06ca3bb02fc57655312090e6
7af5f9f12afe0f5030c1266d0adebf73377b062d50ef6e0bcaccab6da@[::]:30302"
>
```

Figure 14. Check peer-to-peer network between node 1 and node 2

Conclusion

In this tutorial, we have completed creating the local private multi-node Ethereum network (peer-to-peer network). However, this is also a very important part of the ethereum protocol because blockchain is based on peer-to-peer network.

Reference

[1] https://codeburst.io/build-your-first-ethereum-smart-contract-with-solidity-tutorial-94171d6b1c4b

[2] https://btcblockchain.wordpress.com/2017/09/27/how-to-run-private-ethereum-blockchain-on-windows/