Setting Up a Private Ethereum Network:

1. Prerequisites:

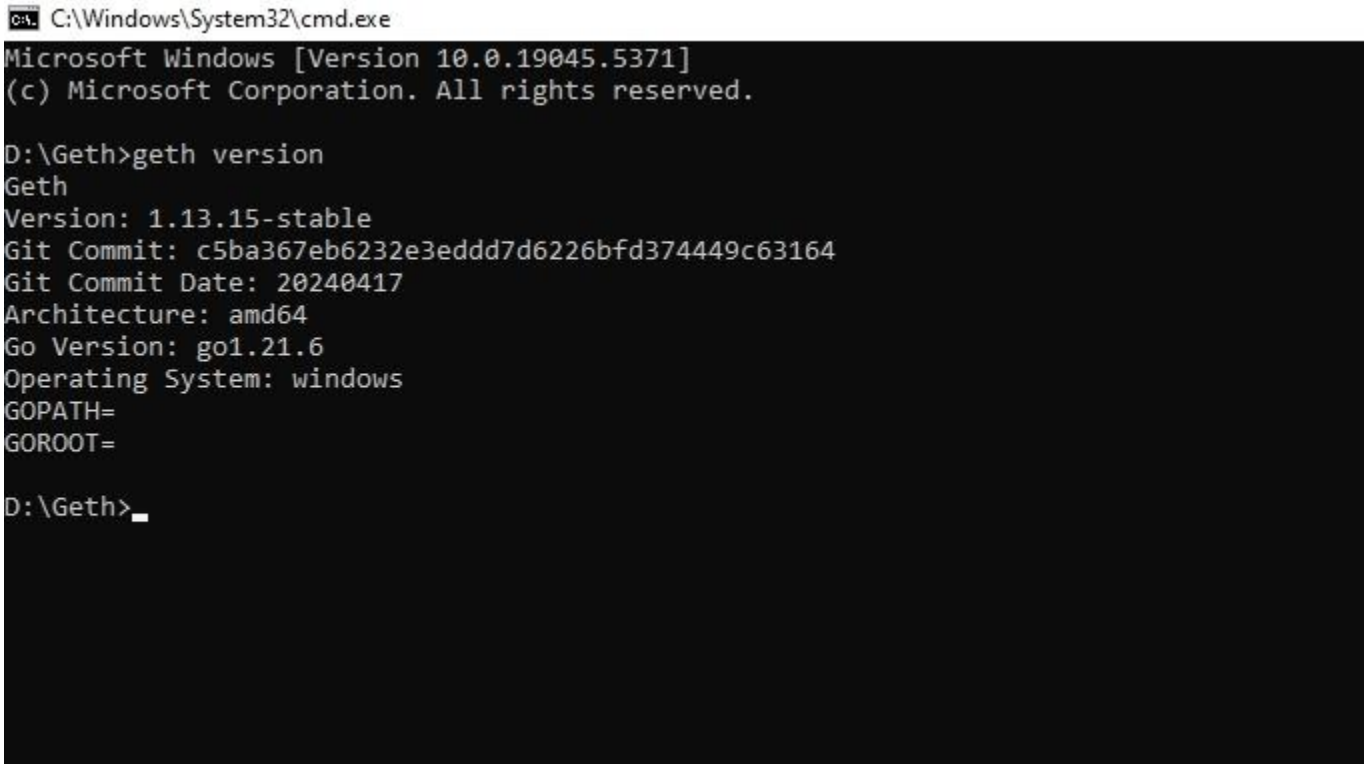
Required Software:

- Geth (Go Ethereum) v1.13.15 or later.
- Windows 10/11 or Linux operating system

Installation Verification:

Cmd:

- Geth version



2. Directory Setup and Account Creation:

Create Project Structure:

Cmd:

- Open cmd from geth setup directory.
- mkdir node1 node2

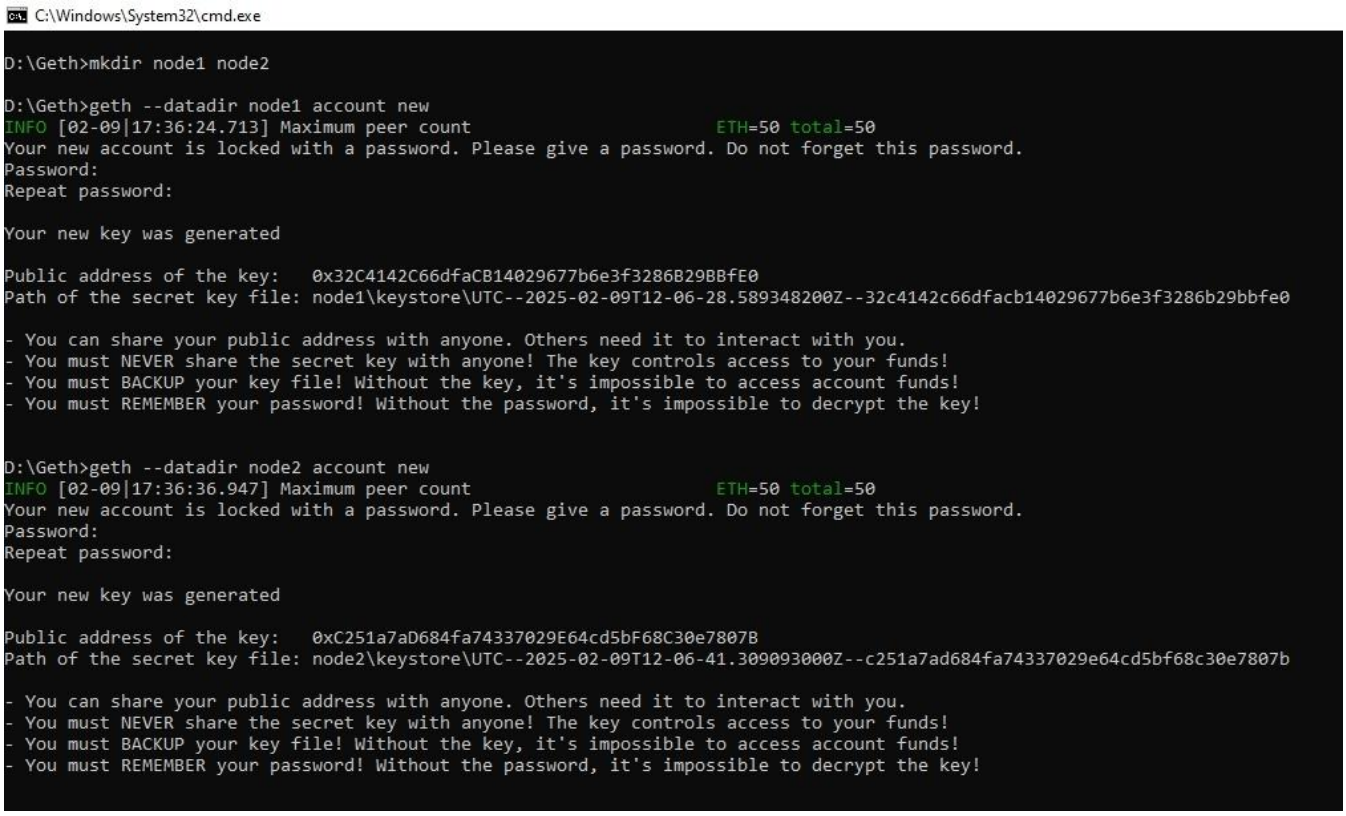
Create Account for Node 1 and Node 2

Cmd:

- geth --datadir node1 account new
- geth --datadir node2 account new

Command Explanation:

- **--datadir:** Specifies the data directory
- **account new:** Creates a new account.



After creating password.txt in both file Node 1 and Node 2 which have your password of both accounts generated at time of accounts creation.

3. Genesis Configuration:

Create a file named genesis.json with the following content:

```
{
  "config": {
```

```
    "chainId": 123454321,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "istanbulBlock": 0,
    "muirGlacierBlock": 0,
    "berlinBlock": 0,
    "londonBlock": 0,
    "arrowGlacierBlock": 0,
    "grayGlacierBlock": 0,
    "clique": {
      "period": 5,
      "epoch": 30000
    }
  },
  "difficulty": "1",
  "gasLimit": "9000000",
  "extradata":
"0x0000000000000000000000000000000000000000000000000000000000000000032C4142C66
dfaCB14029677b6e3f3286B29BBfE0000000000000000000000000000000000000000000000000000
```

0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",

"alloc": {

"32C4142C66dfaCB14029677b6e3f3286B29BBfE0": { "balance": "10000000000000000" },

"C251a7aD684fa74337029E64cd5bF68C30e7807B": { "balance": "10000000000000000" }

}

}

Purpose:

1. **Network Identity**:
    - ChainID uniquely identifies your network
    - Prevents cross-chain replay attacks
    - Distinguishes from other Ethereum network.

2. **Protocol Rules**:
    - Defines consensus algorithm (PoA/Clique in this case)
    - Sets block timing and validation rules
    - Establishes initial state of the blockchain

3. **Initial State**:
    - Allocates initial ether to accounts
    - Sets up initial validators
    - Define both node1 and node2 public address of the key

4. Network Initialization:

Initialize Node 1 and Node 2:

Cmd:

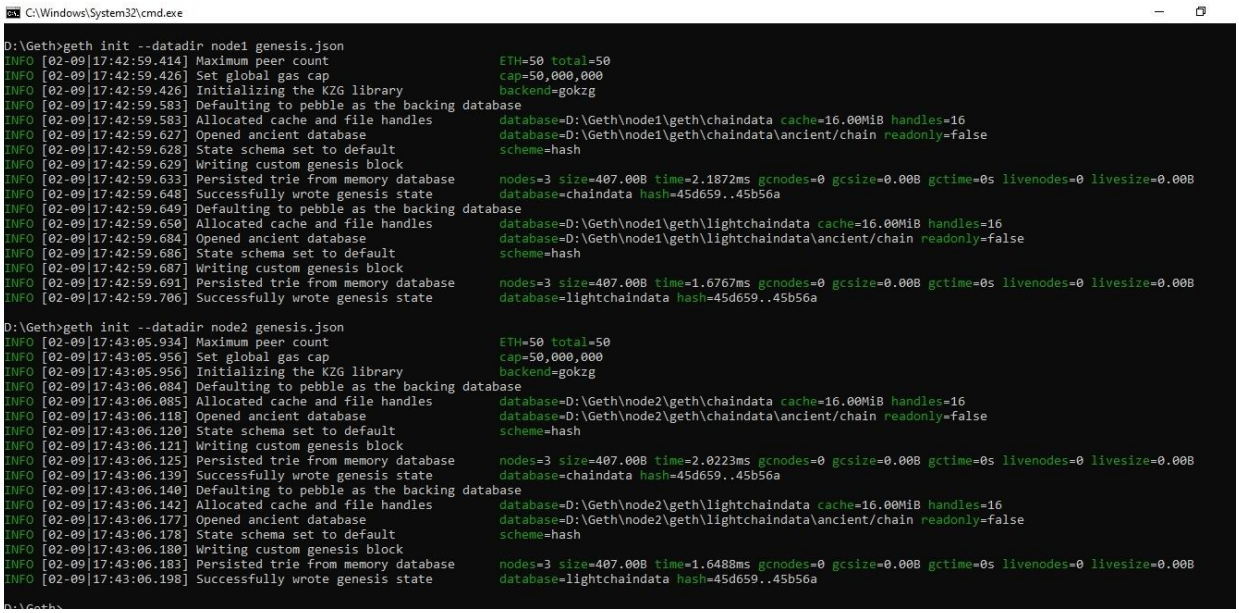
- geth init --datadir node1 genesis.json
- geth init --datadir node2 genesis.json

Command explanation:

- **init**: Initializes new genesis block
- **--datadir**: Specifies the data directory
- **genesis.json**: The genesis configuration file

Purpose:

1. **Blockchain Start**:
    - Creates the first block (genesis)
    - Establishes blockchain state
    - Sets up network parameters
2. **Consistency**:
    - Ensures all nodes start from the same state
    - Maintains network synchronization
    - Prevents fork issues.



5. Bootnode Setup:

Create Bootnode Key:

Cmd:

bootnode -genkey boot.key

Start Bootnode:

Cmd:

bootnode -nodekey boot.key -addr :30335

Command explanation:

- **-nodekey**: Specifies the node's private key file
- **-addr**: Specifies the listening address and port

Purpose:

1. **Node Discovery**: Acts as a meeting point for network participants
2. **Network Organization**: Helps nodes find and connect to each other
3. **Decentralization**: Enables peer-to-peer network formation
4. **Efficiency**: Reduces the need for manual peer addition.

6. Node Startup:

Start Node 1:

Cmd:

geth --datadir node1 --port 30306 --bootnodes enode://fb96930486d0cf4ffb18a0f11a7993a31346004211113ac1127577060e3126299bd1c538dd8de882994399e8b8a490182c0b6e2025d70e400d05eb5176a5bad7@127.0.0.1:0?discport=30335 --networkid 123454321 --unlock 0x32C4142C66dfaCB14029677b6e3f3286B29BBfE0 --password node1/password.txt --authrpc.port 8551 --mine --miner.etherbase 0x32C4142C66dfaCB14029677b6e3f3286B29BBfE0

Start Node 2:

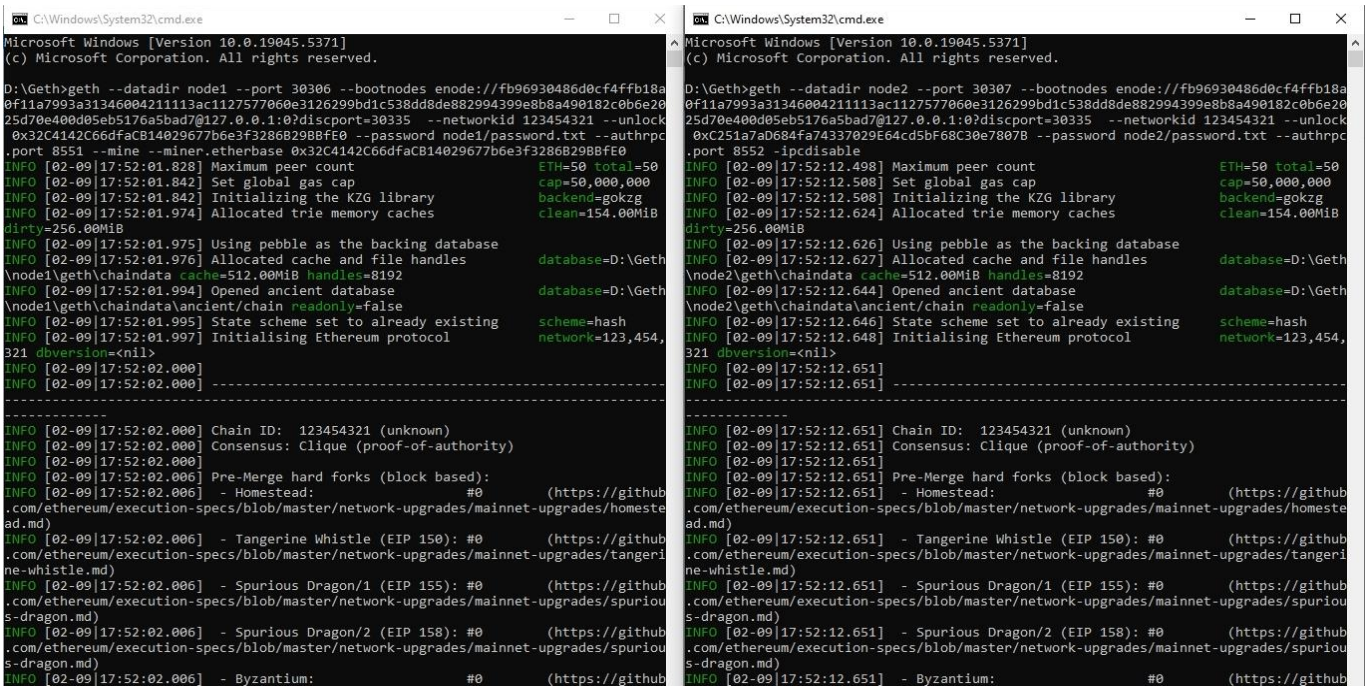
Cmd:

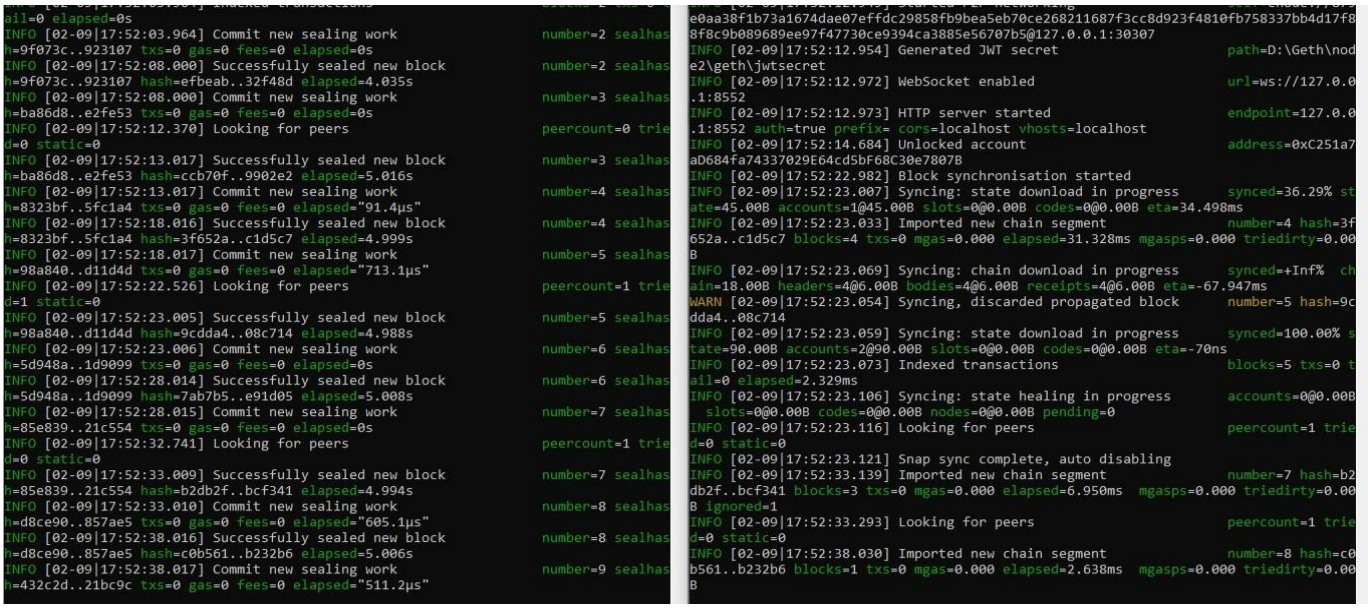
geth --datadir node2 --port 30307 --bootnodes enode://fb96930486d0cf4ffb18a0f11a7993a31346004211113ac1127577060e3126299bd1c538dd8de882994399e8b8a490182c0b6e2025d70e400d05eb5176a5bad7@127.0.0.1:0?discport=30335 --networkid 123454321 --unlock 0xC251a7aD684fa74337029E64cd5bF68C30e7807B --password node2/password.txt --authrpc.port 8552 –ipcdisable

Command parameters explained:

- **--port:** P2P network listening port

- **--bootnode**s: URL of bootnode for peer discovery
- **--networkid:** Unique identifier of the private network
- **--unlock:** Account to unlock for mining/transactions
- **--password:** File containing account password
- **--mine:** Enable mining
- **--miner.etherbase:** Account to receive mining rewards
- **--ipcdisable:** Is used because it's the second node (node2) running on the same machine, preventing any IPC-related conflicts with the first node.





7. Network Testing and Transactions Perform:

Cmd:

```
net.peerCount

eth.getBalance()

eth.sendTransaction()

eth.getTransaction()
```

Purpose:

1. **Connectivity Verification**:

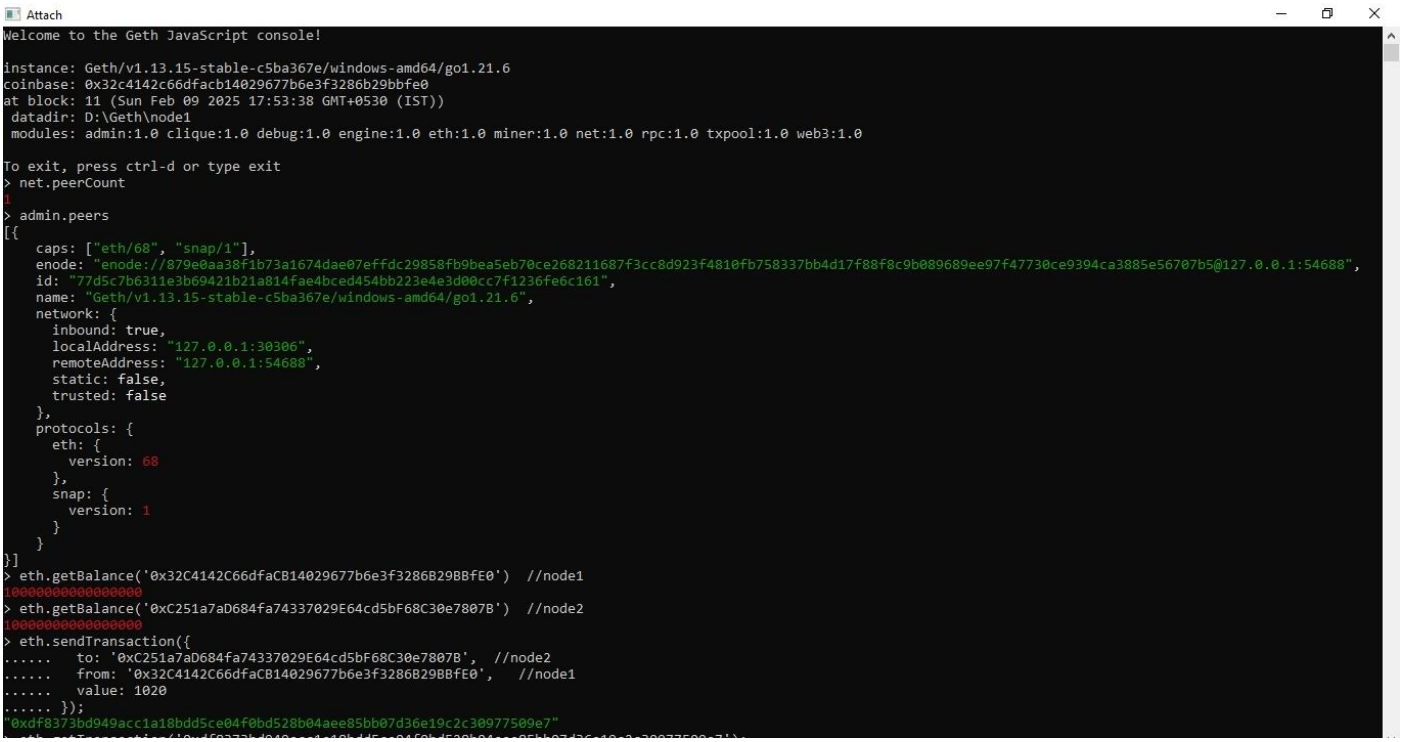
    - Ensures nodes can find each other.
    - Verifies peer connections
    - Checks network stability

2. **Functionality Testing**:

    - Verifies transaction processing
    - Confirms block creation
    - Tests account operations

3. **Performance Validation**:

    - Checks block propagation
    - Verifies consensus mechanism
    - Validates network speed


Attach
Welcome to the Geth JavaScript console!
instance: Geth/v1.13.15-stable-c5ba367e/windows-amd64/go1.21.6
coinbase: 0x32c4142c66dfacb14029677b6e3f3286b29bbfe0
at block: 11 (Sun Feb 09 2025 17:53:38 GMT+0530 (IST))
datadir: D:\Geth\node1
modules: admin:1.0 clique:1.0 debug:1.0 engine:1.0 eth:1.0 miner:1.0 net:1.0 rpc:1.0 txpool:1.0 web3:1.0
To exit, press ctrl-d or type exit
> net.peerCount
1
> admin.peers
[{
caps: ["eth/68", "snap/1"],
enode: "enode://879e0aa38f1b73a1674dae07effdc29858fb9bea5eb70ce268211687f3cc8d923f4810fb758337bb4d17f88f8c9b089689ee97f47730ce9394ca3885e56707b5@127.0.0.1:54688",
id: "77d5c7b6311e3b69421b21a814fae4bced454bb223e4e3d00cc7f1236fe6c161",
name: "Geth/v1.13.15-stable-c5ba367e/windows-amd64/go1.21.6",
network: {
inbound: true,
localAddress: "127.0.0.1:30306",
remoteAddress: "127.0.0.1:54688",
static: false,
trusted: false
},
protocols: {
eth: {
version: 68
},
snap: {
version: 1
}
}
}]
> eth.getBalance('0x32C4142C66dfaCB14029677b6e3f3286B29BBfE0') //node1
1000000000000000
> eth.getBalance('0xC251a7aD684fa74337029E64cd5bF68C30e7807B') //node2
1000000000000000
> eth.sendTransaction({
..... to: '0xC251a7aD684fa74337029E64cd5bF68C30e7807B', //node2
..... from: '0x32C4142C66dfaCB14029677b6e3f3286B29BBfE0', //node1
..... value: 1020
..... });
"0xdf8373bd949acc1a18bdd5ce04f0bd528b04aee85bb07d36e19c2c30977509e7"


Attach
..... value: 1020
..... });
"0xdf8373bd949acc1a18bdd5ce04f0bd528b04aee85bb07d36e19c2c30977509e7"
> eth.getTransaction('0xdf8373bd949acc1a18bdd5ce04f0bd528b04aee85bb07d36e19c2c30977509e7');
{
accessList: [],
blockHash: "0x6d42467659dc805f503d688795ab79c4e3130b4e205787168f3ec57a1f5ddf87",
blockNumber: 117,
chainId: "0x75bc371",
from: "0x32c4142c66dfacb14029677b6e3f3286b29bbfe0",
gas: 21000,
gasPrice: 1000000167,
hash: "0xdf8373bd949acc1a18bdd5ce04f0bd528b04aee85bb07d36e19c2c30977509e7",
input: "0x",
maxFeePerGas: 1000000380,
maxPriorityFeePerGas: 1000000000,
nonce: 0,
r: "0x147baf91545411453b49afd0ec7e9f5739f26f546786aed06ed4d6b35f9ed871",
s: "0x66299cae4c049514c2baac8524fe221860f1488e5df769961403034e0bc5e591",
to: "0xc251a7ad684fa74337029e64cd5bf68c30e7807b",
transactionIndex: 0,
type: "0x2",
v: "0x1",
value: 1020,
yParity: "0x1"
}
> eth.sendTransaction({
to: '0xC251a7aD684fa74337029E64cd5bF68C30e7807B', //node2
from: '0x32C4142C66dfaCB14029677b6e3f3286B29BBfE0', //node1
value: 1020
> eth.getBalance('0x32C4142C66dfaCB14029677b6e3f3286B29BBfE0') //node1
0999999996491980
> eth.sendTransaction({
to: '0xC251a7aD684fa74337029E64cd5bF68C30e7807B', //node2
from: '0x32C4142C66dfaCB14029677b6e3f3286B29BBfE0', //node1
value: 1020
> eth.getBalance('0xC251a7aD684fa74337029E64cd5bF68C30e7807B') //node2
1000000000001020
>