Intro. To Blockchain & Cryptocurrency

Hands On: Creating Own Local Private Ethereum Network

By Swagatika Sahoo

ETHEREUM

WHAT IS ETHEREUM?

- Ethereum is an open platform that enables developers to build and deploy decentralized applications such as smart contracts and other complex legal and financial applications.

- Ethereum is now currently the cryptocurrency with the second highest coin market cap and is expected by some to surpass Bitcoin as both a valued investment and as the world's most popular cryptocurrency.

WHY ETHEREUM?

- Uptime
- Security
- Almost Free
- Transparency
- Micro payments
- DAOs, Consensus applications, governance Identity / Reputation Services

Ethereum Clients

There are several popular clients to work with the Ethereum blockchain. They are:

- **eth** – a client written in C++
- **geth** – a client written in Go
- **pyethapp** – a client written in Python
- **Parity** – a client written in Rust
- **Mantis** – a client written in Scala
- **Harmony** – a client written in Java

...

Lets now discuss more about geth client….

What is Geth?

- Geth is a go-ethereum client multipurpose command line tool that runs a full Ethereum node implemented in Go.

- It offers three interfaces: the command line subcommands and options, a Json-rpc server and an interactive console.

- It will connect to the existing live blockchain or create its own, depending on provided settings.

- By installing and running geth, we can take part in the ethereum frontier live network and we can participating in making the Ethereum network better and stronger.

Ethereum: Setting Up A Private Blockchain(Geth Client)

- Go the Go Ethereum (geth) site and download the binary for your operating system.

Environment Setup

Links for Geth Installation :

- Website: http://ethereum.github.io/go-ethereum/
- GitHub: https://github.com/ethereum/go-ethereum
- Wiki: https://github.com/ethereum/go-ethereum/wiki/geth

Geth Installation…

Goto https://geth.ethereum.org/downloads/ and download geth

Geth help

```
C:\Users\IITP>geth help
NAME:
   geth - the go-ethereum command line interface

   Copyright 2013-2019 The go-ethereum Authors

USAGE:
   geth [options] command [command options] [arguments...]

VERSION:
   1.9.9-stable-01744997

COMMANDS:
   account                          Manage accounts
   attach                           Start an interactive JavaScript environment (connect to node)
   console                          Start an interactive JavaScript environment
   copydb                           Create a local chain from a target chaindata folder
   dump                             Dump a specific block from storage
   dumpconfig                       Show configuration values
   export                           Export blockchain into file
   export-preimages                 Export the preimage database into an RLP stream
   import                           Import a blockchain file
   import-preimages                 Import the preimage database from an RLP stream
   init                             Bootstrap and initialize a new genesis block
   inspect                          Inspect the storage size for each type of data in the database
   js                               Execute the specified JavaScript files
   license                          Display license information
   makecache                        Generate ethash verification cache (for testing)
   makedag                          Generate ethash mining DAG (for testing)
   removedb                         Remove blockchain and state databases
   retesteth                        Launches geth in retesteth mode
   version                          Print version numbers
   wallet                           Manage Ethereum presale wallets
   help, h                          Shows a list of commands or help for one command

ETHEREUM OPTIONS:
  --config value                    TOML configuration file
  --datadir value                   Data directory for the databases and keystore (default: "C:\\Users\\IITP\\AppData\\Roaming\\Ethereum")
  --datadir.ancient value           Data directory for ancient chain segments (default = inside chaindata)
  --keystore value                  Directory for the keystore (default = inside the datadir)
  --nousb                           Disables monitoring for and managing USB hardware wallets
  --pcscdpath value                 Path to the smartcard daemon (pcscd) socket file
  --networkid value                 Network identifier (integer, 1=Frontier, 2=Morden (disused), 3=Ropsten, 4=Rinkeby) (default: 1)
  --testnet                         Ropsten network: pre-configured proof-of-work test network
  --rinkeby                         Rinkeby network: pre-configured proof-of-authority test network
  --goerli                          Görli network: pre-configured proof-of-authority test network
  --syncmode value                  Blockchain sync mode ("fast", "full", or "light") (default: fast)
  --exitwhensynced                  Exits after block synchronisation completes
  --gcmode value                    Blockchain garbage collection mode ("full", "archive") (default: "full")
  --ethstats value                  Reporting URL of a ethstats service (nodename:secret@host:port)
  --identity value                  Custom node name
  --lightkdf                        Reduce key-derivation RAM & CPU usage at some expense of KDF strength
  --whitelist value                 Comma separated block number-to-hash mappings to enforce (<number>=<hash>)

LIGHT CLIENT OPTIONS:
  --light.serve value               Maximum percentage of time allowed for serving LES requests (multi-threaded processing allows values over 100) (default: 0)
  --light.ingress value             Incoming bandwidth limit for serving light clients (kilobytes/sec, 0 = unlimited) (default: 0)
  --light.egress value              Outgoing bandwidth limit for serving light clients (kilobytes/sec, 0 = unlimited) (default: 0)
  --light.maxpeers value            Maximum number of light clients to serve, or light servers to attach to (default: 100)
  --ulc.servers value               List of trusted ultra-light servers
```

Create Custom Ethereum Blockchain

- Create a working folder/directory for this exercise. It will be used to hold binaries and configuration files.
- Create the genesis block
- Create storage of the blockchain
- Deploy blockchain nodes

Genesis block

- This block is the first block in the chain and a json file that stores the configuration of the chain and the only one without a predecessor.
- Create a new project directory. Within it create the **genesis.json** file.

genesis.json

```
{
  "config": {
        "chainId": 4777,
        "homesteadBlock": 0,
        "eip150Block": 0,
        "eip155Block": 0,
        "eip158Block": 0
    },
  "alloc"      : {},
  "difficulty" : "1",
  "extraData"  : "",
  "gasLimit"   : "0x7A1200",
  "parentHash" : "0x0000000000000000000000000000000000000000000000000000000000000000",
  "timestamp"  : "0x00"
}
```

Let me explain the contents of the Genesis file in brief:

- **chainId** – This is the chain identification number that is used to distinguish between Blockchains.

- **homesteadBlock, eip155Block, eip158Block, byzantiumBlock** – These properties are related to chain forking and versioning. We don't need these for now, so let's set them to 0.

- **difficulty** – This number decides how difficult the blocks will be to mine. For Private networks, it's good to set a lower number as it lets you mine blocks quickly, which results in fast transactions.

- **gasLimit** – This number is the total amount of gas that can be used in each block. We don't want our network to hit the limit, so we have set this high.

- **alloc** – This part is used to allocate ethers to already created accounts.

Create the storage of the blockchain

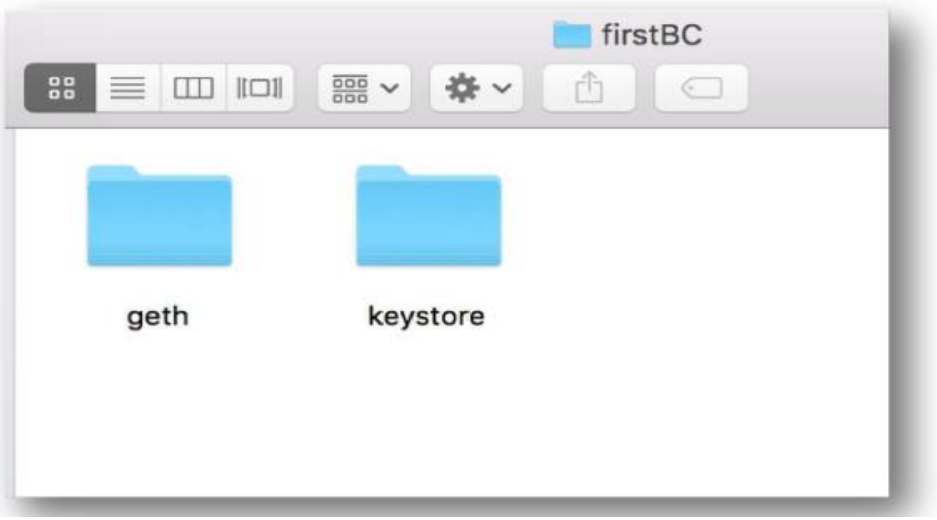
- Once the **genesis.json** file is saved, you are ready to create your first node. To create your first node, open a new terminal window and navigate to your project folder, and type in the following command.

geth –datadir firstBC init genesis.json

```
E:\BlockChain>geth --datadir firstBC init genesis.json
INFO [12-15|19:36:18.168] Maximum peer count                       ETH=50 LES=0 total=50
INFO [12-15|19:36:18.214] Allocated cache and file handles         database=E:\\BlockChain\\firstBC\\geth\\chaindata cac
he=16.00MiB handles=16
INFO [12-15|19:36:18.276] Writing custom genesis block
INFO [12-15|19:36:18.279] Persisted trie from memory database      nodes=0 size=0.00B time=0s gcnodes=0 gcsize=0.00B gct
ime=0s livenodes=1 livesize=0.00B
INFO [12-15|19:36:18.290] Successfully wrote genesis state         database=chaindata                              has
h=567e85…683dd4
INFO [12-15|19:36:18.297] Allocated cache and file handles         database=E:\\BlockChain\\firstBC\\geth\\lightchaindat
a cache=16.00MiB handles=16
INFO [12-15|19:36:18.389] Writing custom genesis block
INFO [12-15|19:36:18.394] Persisted trie from memory database      nodes=0 size=0.00B time=0s gcnodes=0 gcsize=0.00B gct
ime=0s livenodes=1 livesize=0.00B
INFO [12-15|19:36:18.406] Successfully wrote genesis state         database=lightchaindata                         has
h=567e85…683dd4
```

Inside the Blockchain Folder

- **gethfolder:** Store your database.
- **keystore:** Store your Ethereum accounts.



Start the Ethereum peer node

- Start the blockchain through geth console.

- The geth console offers a command line interface with a javascript runtime.

geth --datadir "firstBC" --networkid 1234 --port 11111 "--allow-insecure-unlock“ --nodiscover --ipcdisable console

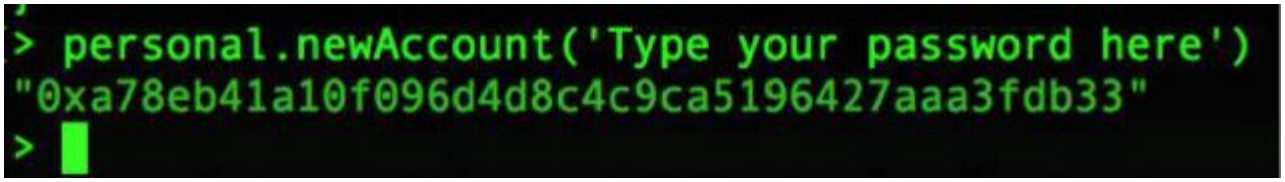
•Networkid provides privacy for your network.

•Other peers joining your network must use the same networkid but port number should be different.

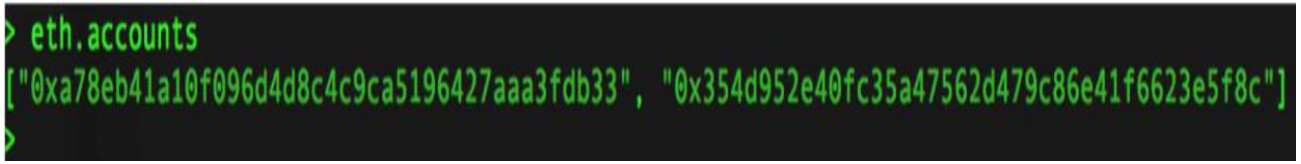
• The console is a Javascript console that lets we send commands to Geth.

Create an account

- Type **personal.newAccount('Type your password here')** to create as many accounts as we need



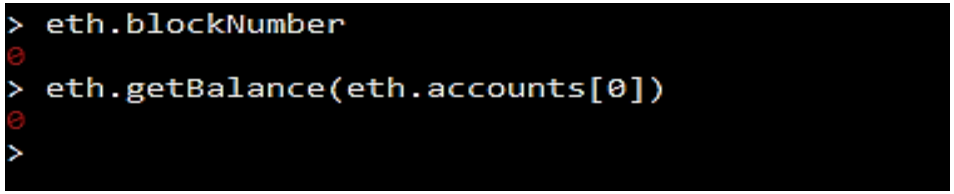
- See the created account(s) by type command **eth.accounts.**



Mining

Before start mining — we have checked current blocks in our private blockchain.

> eth.blockNumber
0
> eth.getBalance(eth.accounts[0])
0



Lets start mining –

Type **miner.start()** to start mining .

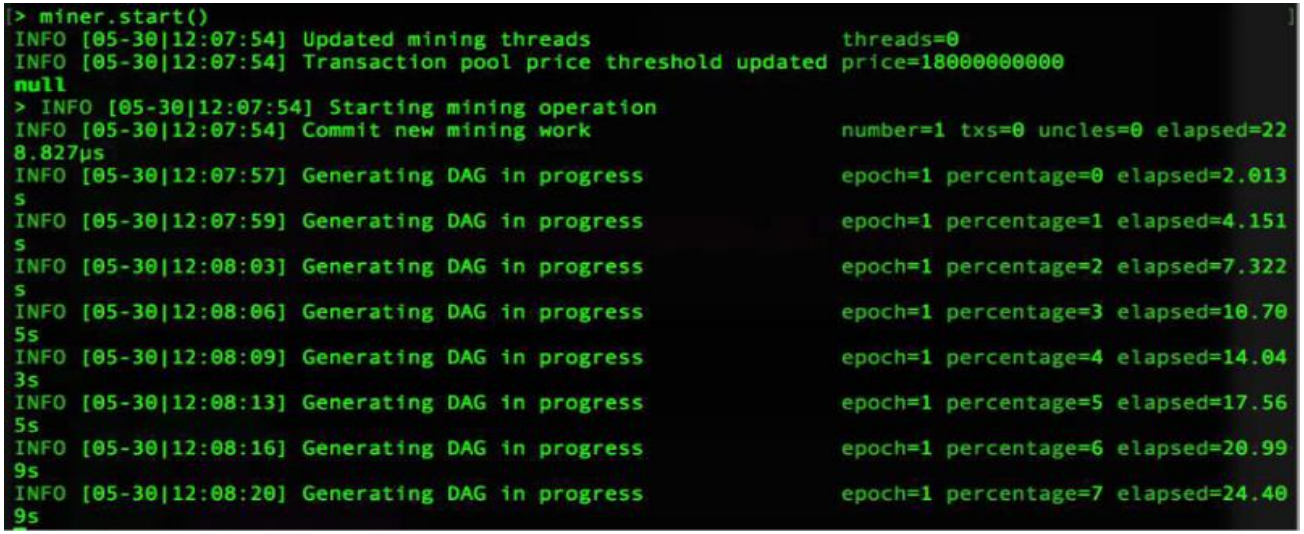
miner.start takes an optional parameter for the number of miner threads.

Once mining is started, let's check balance ether for the number of miner threads.

>acc1 = eth.accounts[0]
>eth.getBalance(acc1)

Mining…

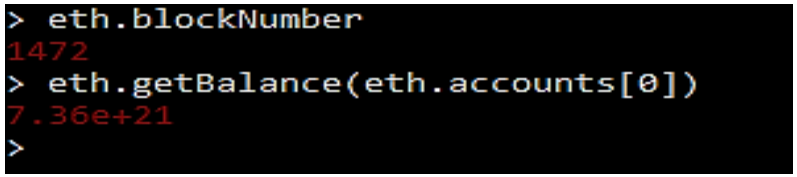
- This will generate the new DAG structure used in ETHash mining, and then start mining blocks.



- Type **miner.stop()** to stop mining

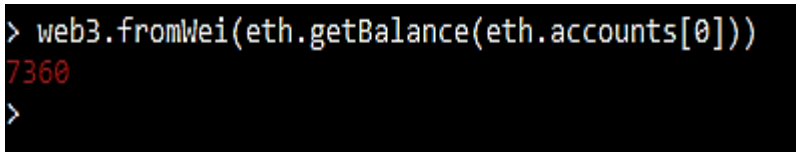
Now check again blocknumber and account balance

- > **eth.blockNumber**
  1472
  > **eth.getBalance(eth.accounts[0])**
  7.36e+21



Balance is showing in wei. We can convert it to ether

- > **web3.fromWei(eth.getBalance(eth.accounts[0]))**
  385



- **Trying to list all accounts —>**

  eth.accounts

  []

```
> eth.accounts
["0xd0c6b97fc25a9aed007ae6478e6468bf69ab9273", "0x20c09560be74b7bdb8c149066563d8663dbd16fe", "0x7e9351d5133ef4e945f2fc77444e7f33363b1b6c"]
```

Now we send ether from one account to other accounts.

> eth.sendTransaction({from:eth.accounts[0], to:eth.accounts[1], value: web3.toWei(10, "ether")})

Error: authentication needed: password or unlock

at web3.js:3143:20

at web3.js:6347:15

at web3.js:5081:36

at <anonymous>:1:1

```
> eth.sendTransaction({from:eth.accounts[0], to:eth.accounts[1], value:23})
Error: authentication needed: password or unlock
    at web3.js:3143:20
    at web3.js:6347:15
    at web3.js:5081:36
    at <anonymous>:1:1
```

- So, before to be able to send transaction we have to unlock sender account.

> **personal.unlockAccount(eth.accounts[0])**

Unlock account 0x4d7287b92bde40e93b0e069d95a2fb829bbd37ef
Passphrase:
true

```
> personal.unlockAccount(eth.accounts[0])
Unlock account 0xd0c6b97fc25a9aed007ae6478e6468bf69ab9273
Passphrase:
true
```

> **eth.sendTransaction({from:eth.accounts[0], to:eth.accounts[1], value: web3.toWei(10, "ether")})**

```
> eth.sendTransaction({from:eth.accounts[0], to:eth.accounts[1], value:23})
INFO [08-21|11:05:59.088] Setting new local account            address=0xD0c6B97FC25a9aED007ae6478e6468bf69aB9273
INFO [08-21|11:05:59.123] Submitted transaction                fullhash=0x0b4ea3f15fb9cbc67373460221c4bd01fde71cb25da840e9671d8a5eb0e7eff5 recipient=0x20c09560BE74b7BDb8C149066563D8663Dbd16FE
"0x0b4ea3f15fb9cbc67373460221c4bd01fde71cb25da840e9671d8a5eb0e7eff5"
```

Lets check balance of the receiver ->

> **eth.getBalance(to:eth.accounts[1])**

0

```
> eth.getBalance(eth.accounts[1])
0
```

- It is marked as a pending transaction -

> eth.pendingTransactions

[{
blockHash: null,
blockNumber: null,
from: “0x4d7287b92bde40e93b0e069d95a2fb829bbd37ef”,
gas: 90000,
gasPrice: 18000000000,
hash: “0x8e38615caa6060ce3a358b3cf136d203e41fdbc6696f8ac14bfae8de1fb1d42f”,
input: “0x”,
nonce: 1,
r: “0x45e3eb16ef1d860272225edd69e3c92fa4e85e56e0caf3173fdca4bbd1c1fc7f”,
s: “0x362f9c9843d5ed21a8948e30726239885bc949c242f127bd059a7fcbc3e9279”,
to: “0x92f8a35f604503c9ad23f6be365d672cf7aebb6b”,
transactionIndex: 0,
v: “0x6096”,
value: 1000000000000000000
}]

```
> eth.pendingTransactions
[{
    blockHash: null,
    blockNumber: null,
    from: "0xd0c6b97fc25a9aed007ae6478e6468bf69ab9273",
    gas: 90000,
    gasPrice: 1000000000,
    hash: "0x0b4ea3f15fb9cbc67373460221c4bd01fde71cb25da840e9671d8a5eb0e7eff5",
    input: "0x",
    nonce: 0,
    r: "0x71bdf5f530cca4fa94f243f08fed1f7f42bbf3f6985e8b30d4d99b73f2c90581",
    s: "0xbd5e55dc059ca1dcc586fa82bb52211aabd62f0ef83f98e6623bf656eb065bc",
    to: "0x20c09560be74b7bdb8c149066563d8663dbd16fe",
    transactionIndex: 0,
    v: "0x42",
    value: 23
}]
```

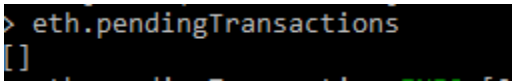
So, we have to start mining and lets see what happens -

> miner.start()

Null

> eth.pendingTransactions

[]



Adding More Peers/Nodes in one system

- Now let's setup a second node in the blockchain network. The process will be similar to setting up Node1.

- Open a new terminal window and navigate to the project folder that contains the **genesis.json** file.

- Initialize the new node with the following command:

  geth --datadir "secondBC" --networkid 1234 --port 11112 "--allow-insecure-unlock" --nodiscover --ipcdisable console

Three important things to note here.

- You must use the same **genesis.json** file.
- You must use a different datadir folder.
- Ensure you use the same network id and port should be different.

- In the console of the second node (you can use either), run **admin.nodeInfo.enode** . You should get something similar to this.

```
> admin.nodeInfo.enode
"enode://800cd2fb9d88744e39c7d458afbb5ae07055b4a0256005b9ca55027d8d25512e619b9eb2f0724b0c364a6a14ad71fb4f78d1ad1dc7385d7e13561edabe446bb9@127.0.0.1:11112?discport=0"
```

- Copy the value of the enode property and in the console of the first node run.

```
> admin.addPeer("enode://800cd2fb9d88744e39c7d458afbb5ae07055b4a0256005b9ca55027d8d25512e619b9eb2f0724b0c364a6a14ad71fb4f78d1ad1dc7385d7e13561edabe446bb9@127.0.0.1:11112?discport=0")
true
> admin.peers
[{
    caps: ["eth/63"],
    enode: "enode://800cd2fb9d88744e39c7d458afbb5ae07055b4a0256005b9ca55027d8d25512e619b9eb2f0724b0c364a6a14ad71fb4f78d1ad1dc7385d7e13561edabe446bb9@127.0.0.1:11112?discport=0",
    id: "1bab50e8c3cd90c822c3cf062c74ea87716ba481acff763c88fa9c22482c8d68",
    name: "Geth/v1.8.23-stable-c9427004/windows-amd64/go1.11.5",
    network: {
      inbound: false,
      localAddress: "127.0.0.1:55590",
      remoteAddress: "127.0.0.1:11112",
      static: true,
      trusted: false
    },
    protocols: {
      eth: {
        difficulty: 20,
        head: "0x5270472fe827114dfec6eb5e6fe8be179ba1cc229da6cedac472f4d0d5aa71f9",
        version: 63
      }
    }
}]
```

Adding new nodes with other systems

- Initialize the data directory on a new system with the same **genesis.json** file (because the default is to use the mainnet).

- On the first system, look up its "enode" discovery address in the console by inspecting this variable:
  admin.nodeInfo

- In the console, peer it with the first node by using the "enode" variable where the host part is replaced with the IP address:

admin.addPeer("enode://6fc14916cefae9082d017a9266a4eed4360719838ba656f59c7bbfdeffc7d933bfeceeae0d7cbdc2f82da2129a6524828ac00afa6778413f0cb2633427745d83@10.0.0.216:30303")

Full list of geth commands

> eth.

```
eth._requestManager             eth.getBlockUncleCount          eth.getWork
eth.accounts                    eth.getCode                     eth.hashrate
eth.blockNumber                 eth.getCoinbase                 eth.iban
eth.call                        eth.getCompilers                eth.icapNamereg
eth.coinbase                    eth.getGasPrice                 eth.isSyncing
eth.compile                     eth.getHashrate                 eth.mining
eth.constructor                 eth.getMining                   eth.namereg
eth.contract                    eth.getPendingTransactions      eth.pendingTransactions
eth.defaultAccount              eth.getProtocolVersion          eth.protocolVersion
eth.defaultBlock                eth.getRawTransaction           eth.resend
eth.estimateGas                 eth.getRawTransactionFromBlock  eth.sendIBANTransaction
eth.filter                      eth.getStorageAt                eth.sendRawTransaction
eth.gasPrice                    eth.getSyncing                  eth.sendTransaction
eth.getAccounts                 eth.getTransaction              eth.sign
eth.getBalance                  eth.getTransactionCount         eth.signTransaction
eth.getBlock                    eth.getTransactionFromBlock     eth.submitTransaction
eth.getBlockNumber              eth.getTransactionReceipt       eth.submitWork
eth.getBlockTransactionCount    eth.getUncle                    eth.syncing
```

> personal.

```
personal._requestManager personal.getListWallets  personal.newAccount
personal.constructor     personal.importRawKey    personal.sendTransaction
personal.deriveAccount   personal.listAccounts    personal.sign
personal.ecRecover       personal.listWallets     personal.unlockAccount
personal.getListAccounts personal.lockAccount
```

Full list of geth commands…

> admin.

```
admin.addPeer               admin.importChain          admin.startRPC
admin.constructor           admin.isPrototypeOf        admin.startWS
admin.datadir               admin.nodeInfo             admin.stopRPC
admin.exportChain           admin.peers                admin.stopWS
admin.getDatadir            admin.propertyIsEnumerable admin.toLocaleString
admin.getNodeInfo           admin.removePeer           admin.toString
admin.getPeers              admin.sleep                admin.valueOf
admin.hasOwnProperty        admin.sleepBlocks
```

> miner.

```
miner.constructor           miner.setEtherbase         miner.toLocaleString
miner.getHashrate           miner.setExtra             miner.toString
miner.hasOwnProperty        miner.setGasPrice          miner.valueOf
miner.isPrototypeOf         miner.start
miner.propertyIsEnumerable  miner.stop
```

Thank you