配置环境 Win10

一、 以太坊安装

下载地址: https://ethereum.github.io/go-ethereum/downloads/

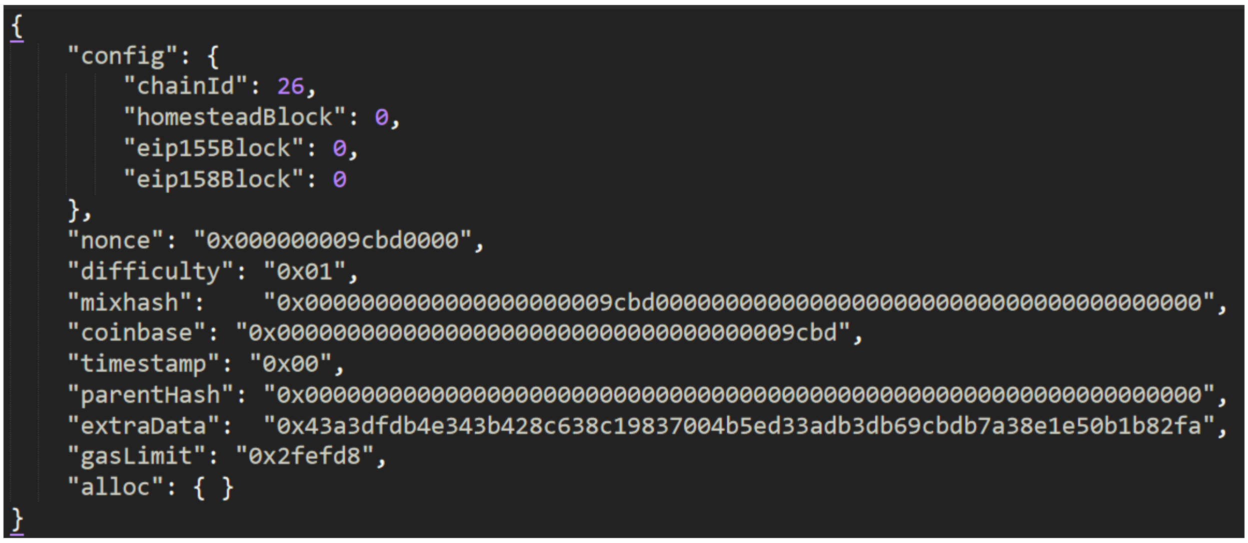
选择

下载后选择目录安装

在 Path 环境变量添加刚刚的安装目录，即可在任意目录使用 geth 了

二、 私有链创世区块搭建

首先要配置 genesis.json 文件，这是为了区分公有链
同一个网络中，创世块必须是一样的，否则无法联通
在准备存放私链的目录下创建创世块文件 genesis.json
我的创世区块配置如下:

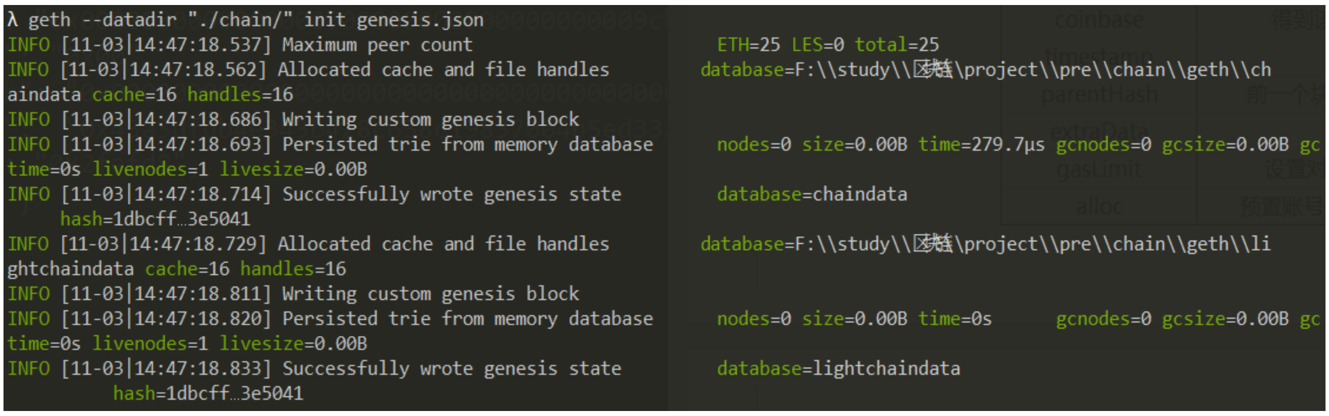


```
{
    "config": {
        "chainId": 26,
        "homesteadBlock": 0,
        "eip155Block": 0,
        "eip158Block": 0
    },
    "nonce": "0x000000009cbd0000",
    "difficulty": "0x01",
    "mixhash":    "0x000000000000000000000009cbd0000000000000000000000000000000000000",
    "coinbase": "0x0000000000000000000000000000000000009cbd",
    "timestamp": "0x00",
    "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
    "extraData":  "0x43a3dfdb4e343b428c638c19837004b5ed33adb3db69cbdb7a38e1e50b1b82fa",
    "gasLimit": "0x2fefd8",
    "alloc": { }
}
```

下面给出一些字段的解释

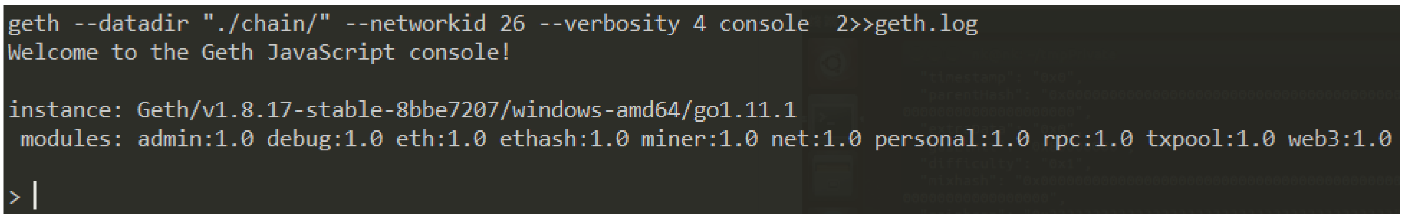
字段	解释
config	用来配置以太坊, 与创世块内容无关
chainID	同一个chainID才能互连, 主链为1 设置其他值来搭建私链
HomesteadBlock	设置为0，表示使用Homestead版本
eip155Block	EIP提案, 接受以防止 replay attacks
eip158Block	EIP提案, 改变处理空账户的方式
nonce	用于挖矿
difficulty	挖矿的难度
mixhash	与nonce配合用于挖矿
coinbase	得到该块奖励的矿工账号
timestamp	时间戳
parentHash	前一个块的Keccak 256哈希值
extraData	额外信息
gasLimit	设置对gas消耗总量的限制
alloc	预置账号以及账号的以太币数量

然后在终端运行 geth --datadir "./chain/" init genesis.json
这会根据你设置的创世块来生成链
运行信息



```
λ geth --datadir "./chain/" init genesis.json
INFO [11-03|14:47:18.537] Maximum peer count                        ETH=25 LES=0 total=25
INFO [11-03|14:47:18.562] Allocated cache and file handles         database=F:\\study\\区块链\project\\pre\\chain\\geth\\ch
aindata cache=16 handles=16
INFO [11-03|14:47:18.686] Writing custom genesis block
INFO [11-03|14:47:18.693] Persisted trie from memory database       nodes=0 size=0.00B time=279.7µs gcnodes=0 gcsize=0.00B gc
time=0s livenodes=1 livesize=0.00B
INFO [11-03|14:47:18.714] Successfully wrote genesis state          database=chaindata
     hash=1dbcff…3e5041
INFO [11-03|14:47:18.729] Allocated cache and file handles         database=F:\\study\\区块链\project\\pre\\chain\\geth\\li
ghtchaindata cache=16 handles=16
INFO [11-03|14:47:18.811] Writing custom genesis block
INFO [11-03|14:47:18.820] Persisted trie from memory database       nodes=0 size=0.00B time=0s      gcnodes=0 gcsize=0.00B gc
time=0s livenodes=1 livesize=0.00B
INFO [11-03|14:47:18.833] Successfully wrote genesis state          database=lightchaindata
          hash=1dbcff…3e5041
```

最后在终端运行 geth --datadir "./chain/" --networkid 26 --verbosity 4 console 2>>geth.log
看到如下界面

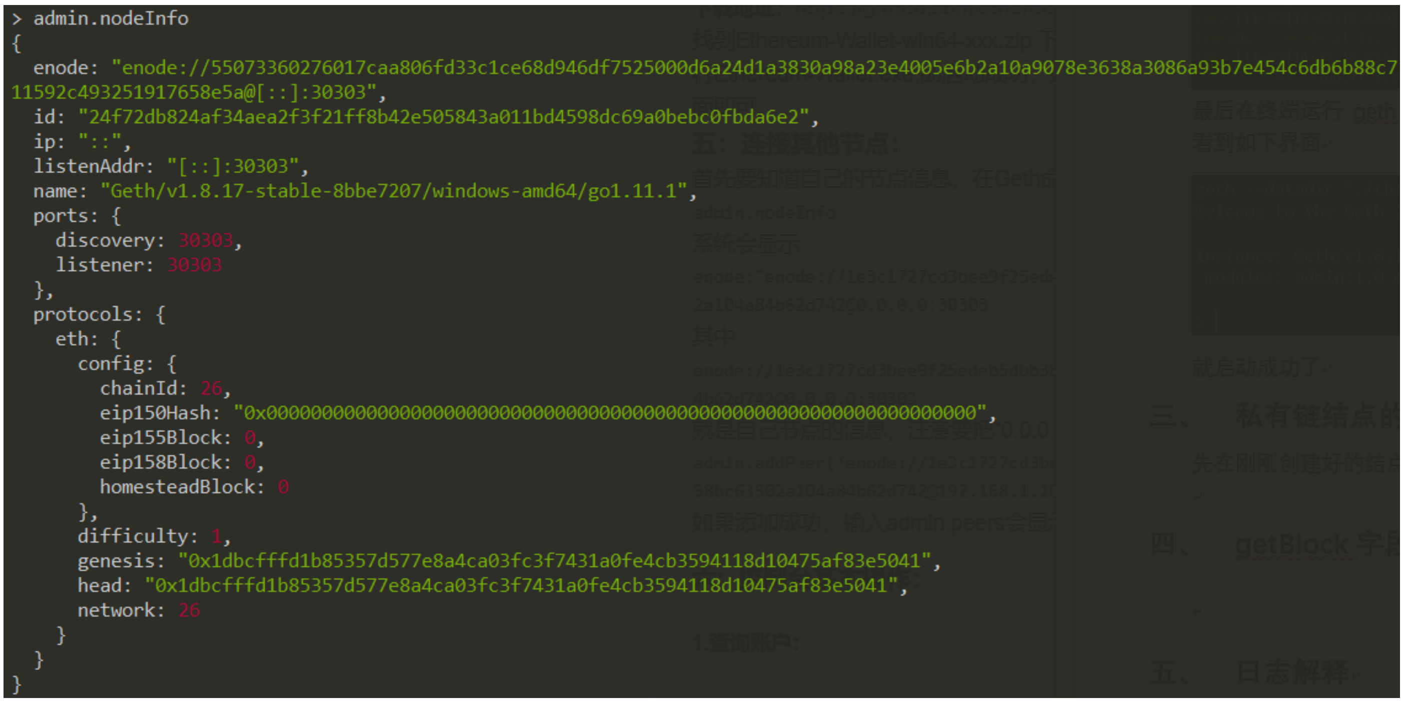


```
geth --datadir "./chain/" --networkid 26 --verbosity 4 console  2>>geth.log
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.17-stable-8bbe7207/windows-amd64/go1.11.1
 modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

> |
```

就启动成功了

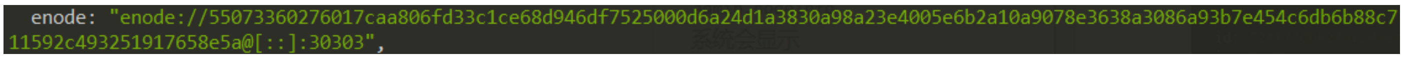
三、 私有链结点的加入

先在刚刚创建好的结点查看结点信息，然后别的结点才能加入，在运行后输入 admin.nodeInfo 查看结点信息



```
> admin.nodeInfo
{
  enode: "enode://55073360276017caa806fd33c1ce68d946df7525000d6a24d1a3830a98a23e4005e6b2a10a9078e3638a3086a93b7e454c6db6b88c7
11592c493251917658e5a@[::]:30303",
  id: "24f72db824af34aea2f3f21ff8b42e505843a011bd4598dc69a0bebc0fbda6e2",
  ip: "::",
  listenAddr: "[::]:30303",
  name: "Geth/v1.8.17-stable-8bbe7207/windows-amd64/go1.11.1",
  ports: {
    discovery: 30303,
    listener: 30303
  },
  protocols: {
    eth: {
      config: {
        chainId: 26,
        eip150Hash: "0x0000000000000000000000000000000000000000000000000000000000000000",
        eip155Block: 0,
        eip158Block: 0,
        homesteadBlock: 0
      },
      difficulty: 1,
      genesis: "0x1dbcfffd1b85357d577e8a4ca03fc3f7431a0fe4cb3594118d10475af83e5041",
      head: "0x1dbcfffd1b85357d577e8a4ca03fc3f7431a0fe4cb3594118d10475af83e5041",
      network: 26
    }
  }
}
```

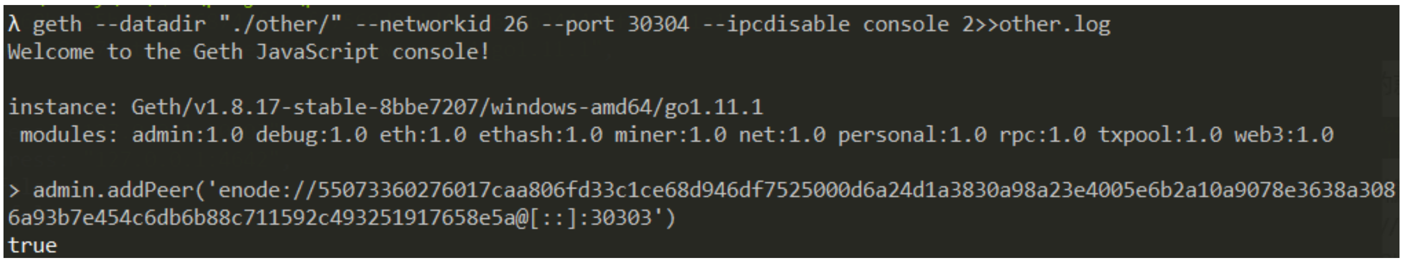
其中, enode 字段显示的就是自己结点的信息



```
  enode: "enode://55073360276017caa806fd33c1ce68d946df7525000d6a24d1a3830a98a23e4005e6b2a10a9078e3638a3086a93b7e454c6db6b88c7
11592c493251917658e5a@[::]:30303",
```

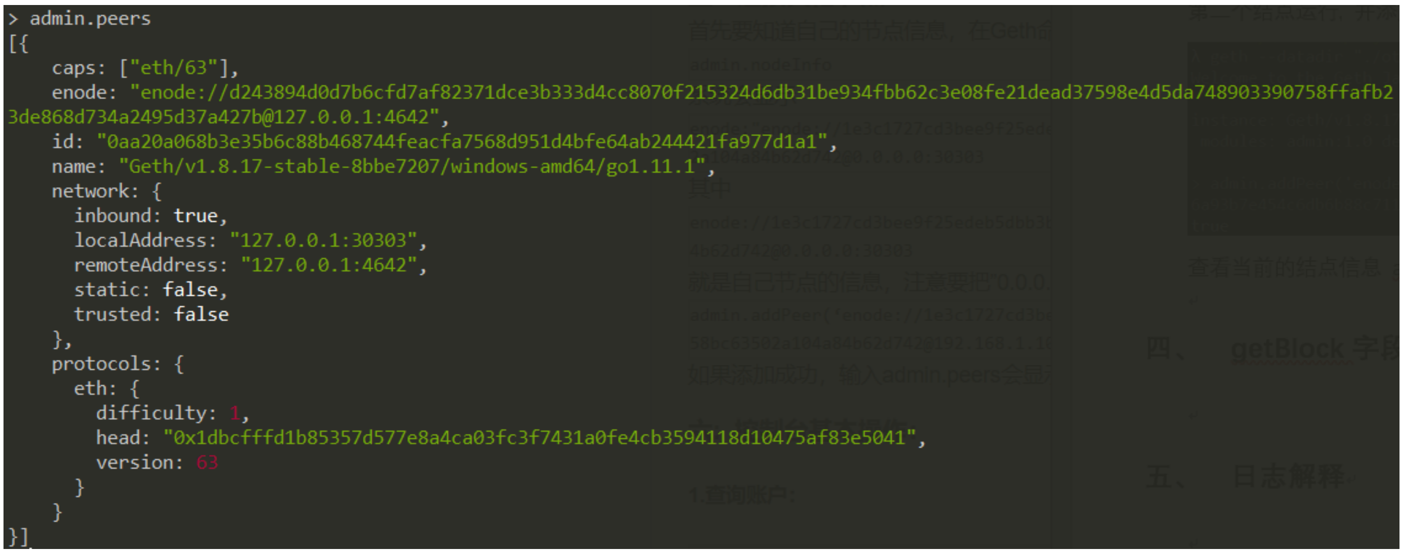
然后在同链的别的结点输入（注意, [::]根据需要换成对应的 ip 地址）
admin.addPeer('enode://55073360276017caa806fd33c1ce68d946df7525000d6a24d1a3830a98a23e4005e6b2a10a9078e3638a3086a93b7e454c6db6b88c711592c493251917658e5a@[::]:30303')
即可加入，下面演示一下本地第二个结点加入

第二个结点运行，并添加结点



```
λ geth --datadir "./other/" --networkid 26 --port 30304 --ipcdisable console 2>>other.log
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.17-stable-8bbe7207/windows-amd64/go1.11.1
 modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

> admin.addPeer('enode://55073360276017caa806fd33c1ce68d946df7525000d6a24d1a3830a98a23e4005e6b2a10a9078e3638a308
6a93b7e454c6db6b88c711592c493251917658e5a@[::]:30303')
true
```

查看当前的结点信息 admin.peers

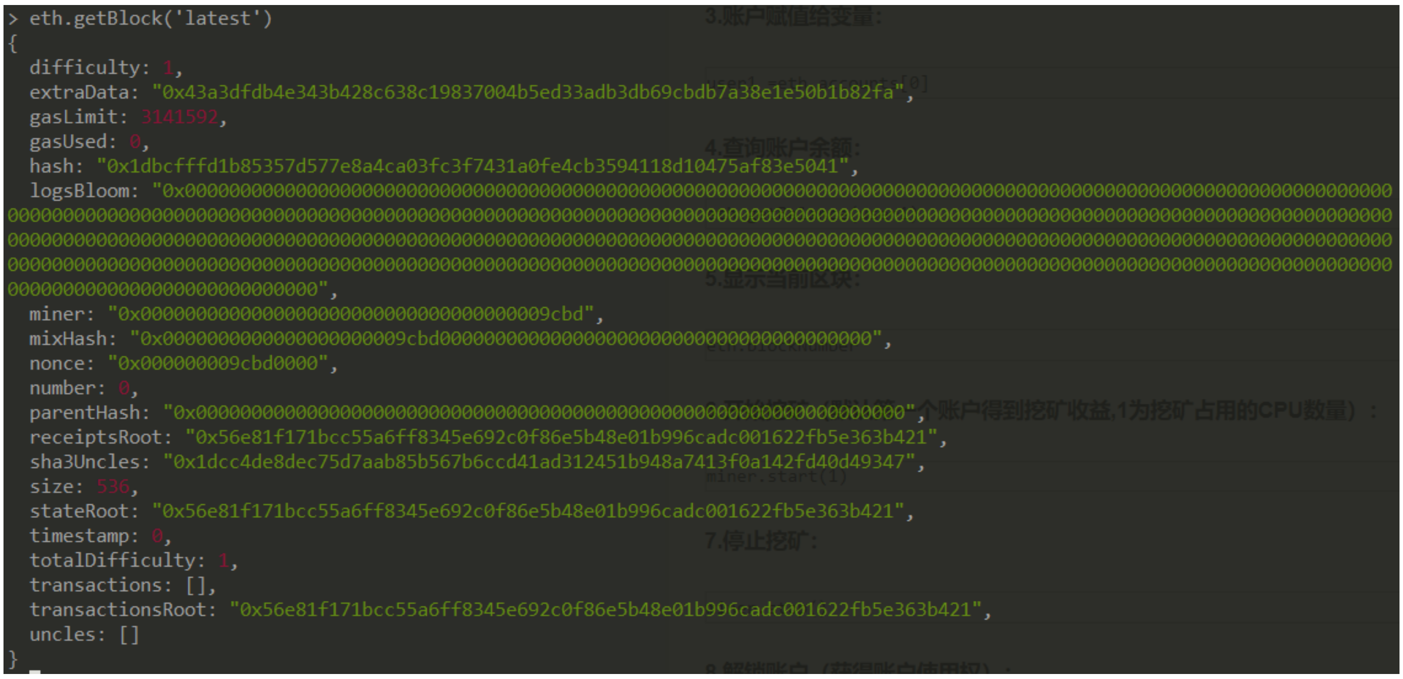


```
> admin.peers
[{
    caps: ["eth/63"],
    enode: "enode://d243894d0d7b6cfd7af82371dce3b333d4cc8070f215324d6db31be934fbb62c3e08fe21dead37598e4d5da748903390758ffafb2
3de868d734a2495d37a427b@127.0.0.1:4642",
    id: "0aa20a068b3e35b6c88b468744feacfa7568d951d4bfe64ab244421fa977d1a1",
    name: "Geth/v1.8.17-stable-8bbe7207/windows-amd64/go1.11.1",
    network: {
      inbound: true,
      localAddress: "127.0.0.1:30303",
      remoteAddress: "127.0.0.1:4642",
      static: false,
      trusted: false
    },
    protocols: {
      eth: {
        difficulty: 1,
        head: "0x1dbcfffd1b85357d577e8a4ca03fc3f7431a0fe4cb3594118d10475af83e5041",
        version: 63
      }
    }
}]
```

这就是刚刚我们新加的结点

四、 getBlock 字段解释

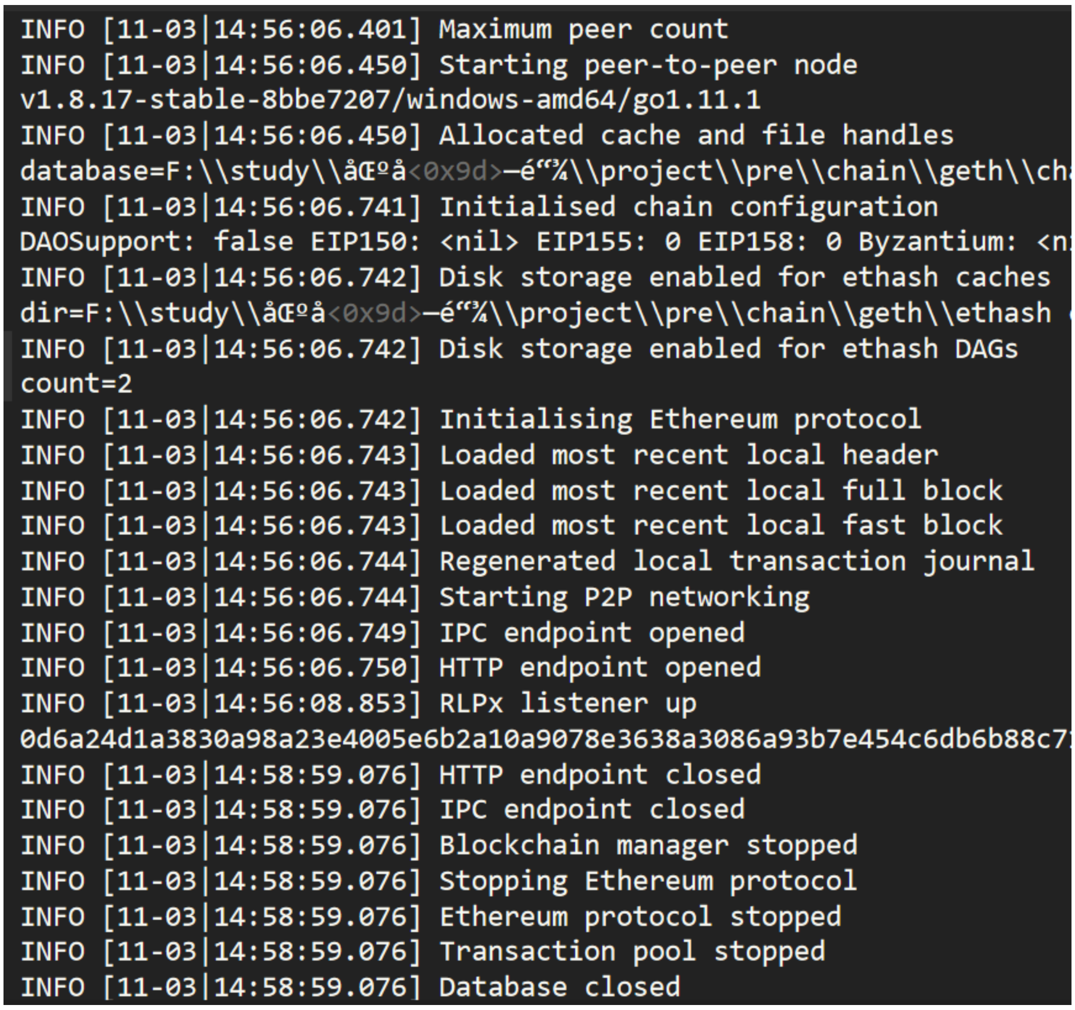
一个 Block 如下



```
> eth.getBlock('latest')
{
  difficulty: 1,
  extraData: "0x43a3dfdb4e343b428c638c19837004b5ed33adb3db69cbdb7a38e1e50b1b82fa",
  gasLimit: 3141592,
  gasUsed: 0,
  hash: "0x1dbcfffd1b85357d577e8a4ca03fc3f7431a0fe4cb3594118d10475af83e5041",
  logsBloom: "0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000",
  miner: "0x0000000000000000000000000000000000009cbd",
  mixHash: "0x0000000000000000000009cbd000000000000000000000000000000000000000",
  nonce: "0x000000009cbd0000",
  number: 0,
  parentHash: "0x0000000000000000000000000000000000000000000000000000000000000000",
  receiptsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  sha3Uncles: "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",
  size: 536,
  stateRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  timestamp: 0,
  totalDifficulty: 1,
  transactions: [],
  transactionsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  uncles: []
}
```

字段	解释
difficulty	挖矿的难度
extraData	额外信息
gasLimit	允许最大的gas
gasUsed	已用的gas
hash	该块的hash值
logsBloom	日志Bloom过滤器
miner	挖出该块的矿工
mixHash	用于挖矿
nonce	PoW生成的hash
number	区块号
parentHash	前一个块的hash值
receiptsRoot	收据树的树根hash
sha3Uncles	uncles的SHA3
size	该块的大小
stateRoot	最终状态的树的树根hash
timestamp	时间戳
totalDifficulty	从创世到当前块的总难度
transactions	包含的交易
transactionsRoot	交易树的树根hash
uncles	树哈希的数组

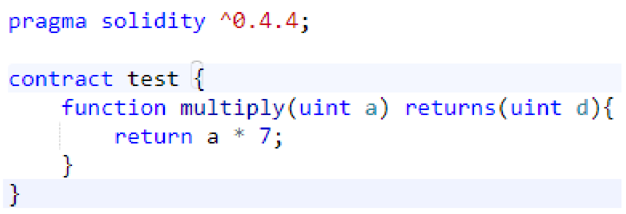
五、　日志解释


```
INFO [11-03|14:56:06.401] Maximum peer count
INFO [11-03|14:56:06.450] Starting peer-to-peer node
v1.8.17-stable-8bbe7207/windows-amd64/go1.11.1
INFO [11-03|14:56:06.450] Allocated cache and file handles
database=F:\\study\\åŒºå<0x9d>—é“¾\\project\\pre\\chain\\geth\\ch
INFO [11-03|14:56:06.741] Initialised chain configuration
DAOSupport: false EIP150: <nil> EIP155: 0 EIP158: 0 Byzantium: <n
INFO [11-03|14:56:06.742] Disk storage enabled for ethash caches
dir=F:\\study\\åŒºå<0x9d>—é“¾\\project\\pre\\chain\\geth\\ethash
INFO [11-03|14:56:06.742] Disk storage enabled for ethash DAGs
count=2
INFO [11-03|14:56:06.742] Initialising Ethereum protocol
INFO [11-03|14:56:06.743] Loaded most recent local header
INFO [11-03|14:56:06.743] Loaded most recent local full block
INFO [11-03|14:56:06.743] Loaded most recent local fast block
INFO [11-03|14:56:06.744] Regenerated local transaction journal
INFO [11-03|14:56:06.744] Starting P2P networking
INFO [11-03|14:56:06.749] IPC endpoint opened
INFO [11-03|14:56:06.750] HTTP endpoint opened
INFO [11-03|14:56:08.853] RLPx listener up
0d6a24d1a3830a98a23e4005e6b2a10a9078e3638a3086a93b7e454c6db6b88c7
INFO [11-03|14:58:59.076] HTTP endpoint closed
INFO [11-03|14:58:59.076] IPC endpoint closed
INFO [11-03|14:58:59.076] Blockchain manager stopped
INFO [11-03|14:58:59.076] Stopping Ethereum protocol
INFO [11-03|14:58:59.076] Ethereum protocol stopped
INFO [11-03|14:58:59.076] Transaction pool stopped
INFO [11-03|14:58:59.076] Database closed
```

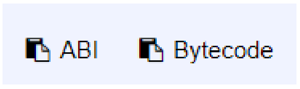
最大结点数；开启 p2p 结点；分配缓存空间和文件句柄；初始化链配置；硬盘空间足够给 ethash 缓存；硬盘空间足够给 ethash 有向无环图；初始化以太坊协议；读取最近本地头；读取最近满块；读取最近快块；重新生成本地交易日志；开始 P2P 网络; IPC 端开启; HTPP 端开启; RLPx 监听器启动; HTTP 端关闭; IPC 端关闭；区块链管理器停止；关闭以太坊协议；以太坊协议停止；交易池关闭；数据库关闭；

六、 智能合约的简单使用

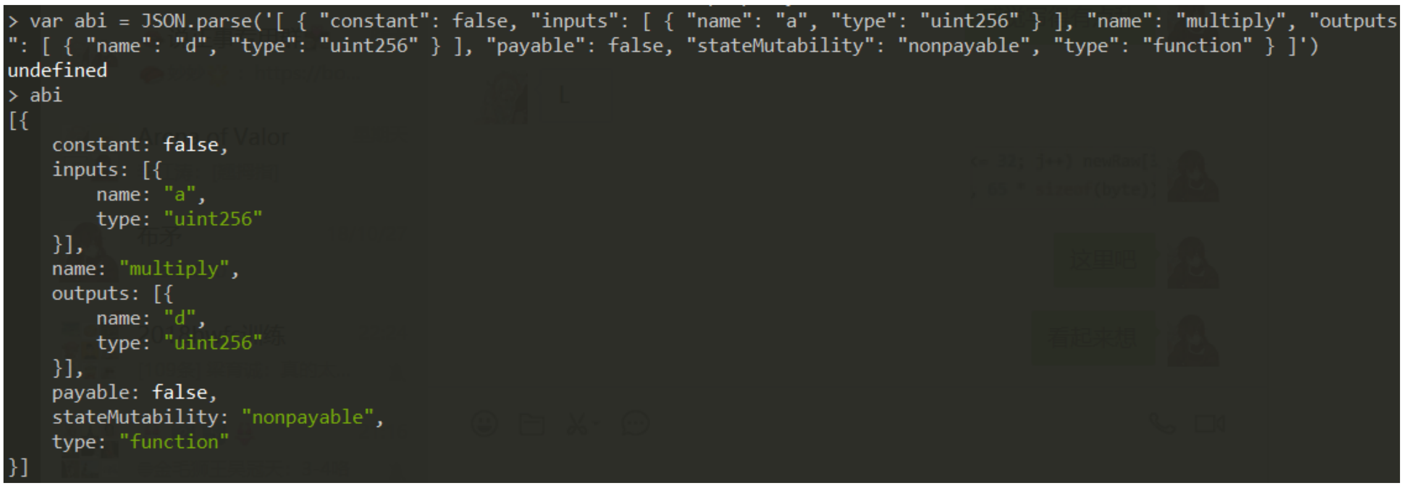
先写一个合约



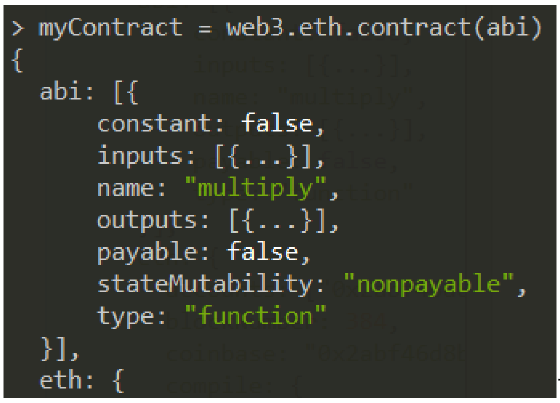
```
pragma solidity ^0.4.4;

contract test {
    function multiply(uint a) returns(uint d){
        return a * 7;
    }
}
```

拷贝字节码和 ABI 

在 bejson 中把 ABI 转义成字符串 http://www.bejson.com
在 JavaScript 执行环境中通过 ABI 创建合约对象



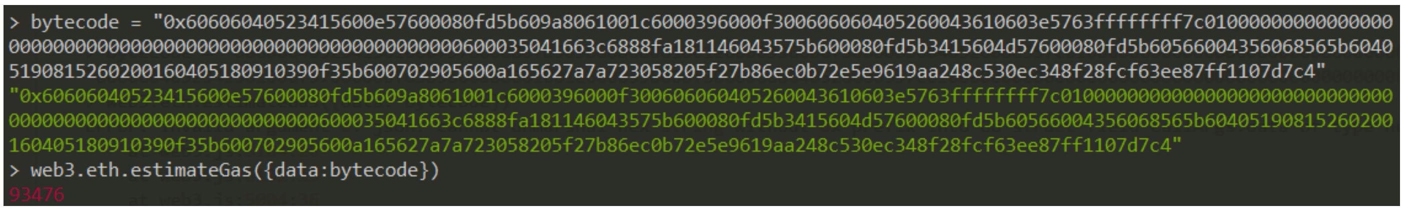
```
> var abi = JSON.parse('[ { "constant": false, "inputs": [ { "name": "a", "type": "uint256" } ], "name": "multiply", "outputs
": [ { "name": "d", "type": "uint256" } ], "payable": false, "stateMutability": "nonpayable", "type": "function" } ]')
undefined
> abi
[{
    constant: false,
    inputs: [{
        name: "a",
        type: "uint256"
    }],
    name: "multiply",
    outputs: [{
        name: "d",
        type: "uint256"
    }],
    payable: false,
    stateMutability: "nonpayable",
    type: "function"
}]
```



```
> myContract = web3.eth.contract(abi)
{
  abi: [{
      constant: false,
      inputs: [{...}],
      name: "multiply",
      outputs: [{...}],
      payable: false,
      stateMutability: "nonpayable",
      type: "function"
  }],
  eth: {
```

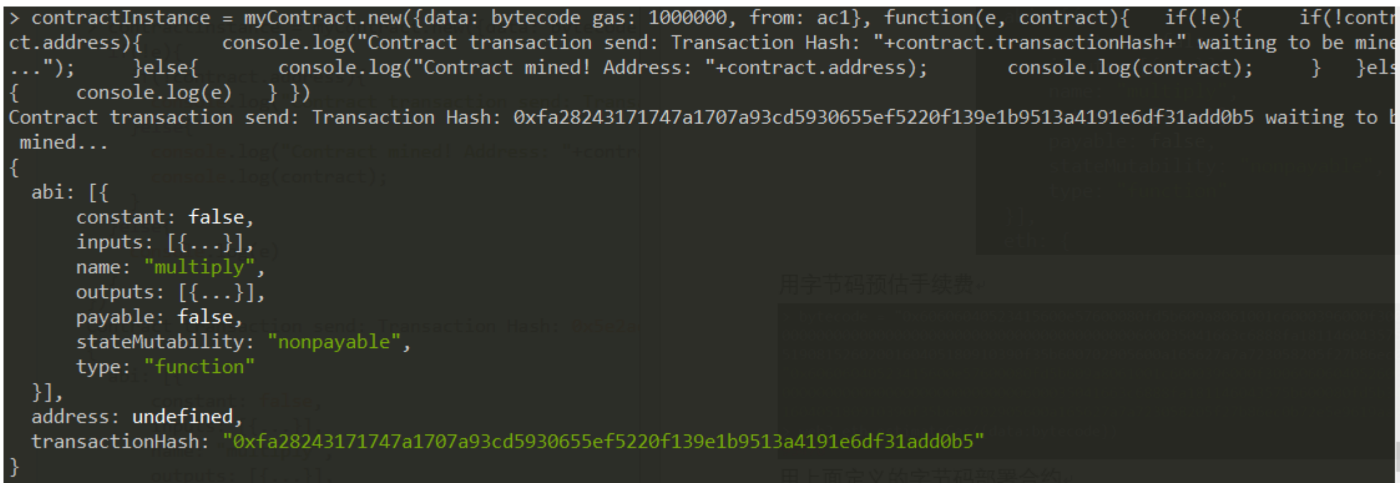
下面还有很多属性略过

用字节码预估手续费



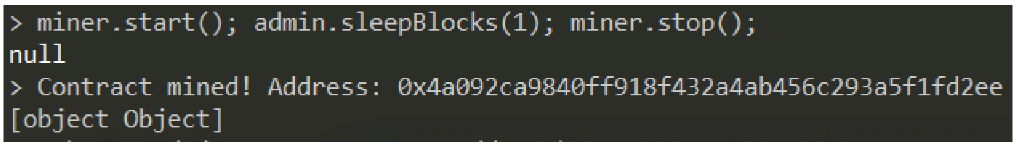
```
> bytecode = "0x60606040523415600e57600080fd5b609a8061001c6000396000f300606060405260043610603e5763ffffffff7c0100000000000000
0000000000000000000000000000000000000000600035041663c6888fa181146043575b600080fd5b3415604d57600080fd5b60566004356068565b6040
519081526020016040518091039 0f35b600702905600a165627a7a723058205f27b86ec0b72e5e9619aa248c530ec348f28fcf63ee87ff1107d7c4"
"0x60606040523415600e57600080fd5b609a8061001c6000396000f300606060405260043610603e5763ffffffff7c0100000000000000000000000000
0000000000000000000000000000600035041663c6888fa181146043575b600080fd5b3415604d57600080fd5b60566004356068565b60405190815260200
16040518091039 0f35b600702905600a165627a7a723058205f27b86ec0b72e5e9619aa248c530ec348f28fcf63ee87ff1107d7c4"
> web3.eth.estimateGas({data:bytecode})
93476
```

在账户余额足够的情况下(不够请挖矿)，先解锁账号 personal.unlockAccount()
然后用上面定义的字节码部署合约



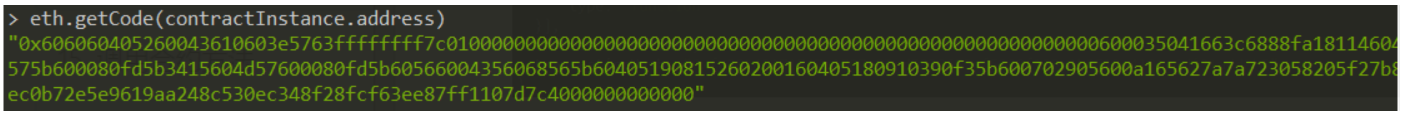
```
> contractInstance = myContract.new({data: bytecode gas: 1000000, from: ac1}, function(e, contract){   if(!e){     if(!contr
ct.address){     console.log("Contract transaction send: Transaction Hash: "+contract.transactionHash+" waiting to be mine
...");     }else{      console.log("Contract mined! Address: "+contract.address);       console.log(contract);     }   }els
{     console.log(e)   } })
Contract transaction send: Transaction Hash: 0xfa28243171747a1707a93cd5930655ef5220f139e1b9513a4191e6df31add0b5 waiting to b
 mined...
{
  abi: [{
      constant: false,
      inputs: [{...}],
      name: "multiply",
      outputs: [{...}],
      payable: false,
      stateMutability: "nonpayable",
      type: "function"
  }],
  address: undefined,
  transactionHash: "0xfa28243171747a1707a93cd5930655ef5220f139e1b9513a4191e6df31add0b5"
}
```

挖矿让合约部署到链上



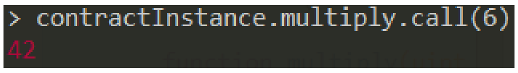
```
> miner.start(); admin.sleepBlocks(1); miner.stop();
null
> Contract mined! Address: 0x4a092ca9840ff918f432a4ab456c293a5f1fd2ee
[object Object]
```

查看合约是否部署成功



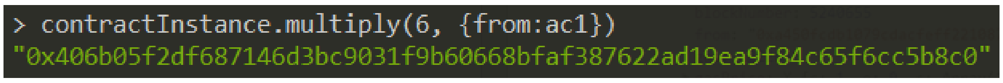
```
> eth.getCode(contractInstance.address)
"0x6060604052600436106603e5763ffffffff7c0100000000000000000000000000000000000000000000000000000000600035041663c6888fa18114604
575b600080fd5b3415604d57600080fd5b605660043560685565b60405190815260200160405180910390f35b600702905600a165627a7a723058205f27b8
ec0b72e5e9619aa248c530ec348f28fcf63ee87ff1107d7c4000000000000"
```

调用合约方法(不修改值，可以用 call 调用)



```
> contractInstance.multiply.call(6)
42
```

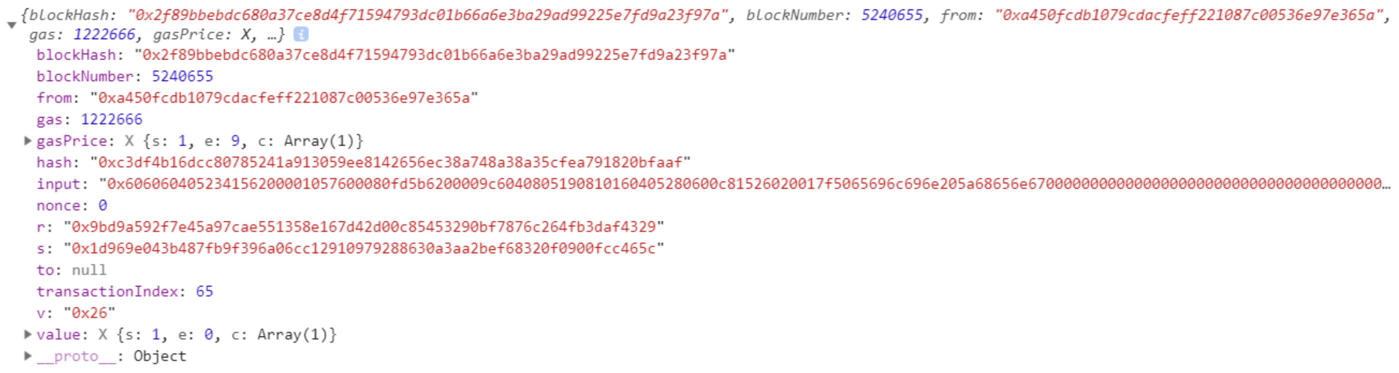
调用会修改合约定义值的则要



```
> contractInstance.multiply(6, {from:ac1})
"0x406b05f2df687146d3bc9031f9b60668bfaf387622ad19ea9f84c65f6cc5b8c0"
```

此时要花费 gas，还需要挖矿等待交易被接受

七、　交易字段的解释



```
▼{blockHash: "0x2f89bbebdc680a37ce8d4f71594793dc01b66a6e3ba29ad99225e7fd9a23f97a", blockNumber: 5240655, from: "0xa450fcdb1079cdacfeff221087c00536e97e365a",
  gas: 1222666, gasPrice: X, …}
   blockHash: "0x2f89bbebdc680a37ce8d4f71594793dc01b66a6e3ba29ad99225e7fd9a23f97a"
   blockNumber: 5240655
   from: "0xa450fcdb1079cdacfeff221087c00536e97e365a"
   gas: 1222666
 ▶ gasPrice: X {s: 1, e: 9, c: Array(1)}
   hash: "0xc3df4b16dcc80785241a913059ee8142656ec38a748a38a35cfea791820bfaaf"
   input: "0x6060604052341562000010576000080fd5b6200009c604080519081016040528060 0c81526020017f5065696c696e205a68656e670000000000000000000000000000000000000…
   nonce: 0
   r: "0x9bd9a592f7e45a97cae551358e167d42d00c85453290bf7876c264fb3daf4329"
   s: "0x1d969e043b487fb9f396a06cc12910979288630a3aa2bef68320f0900fcc465c"
   to: null
   transactionIndex: 65
   v: "0x26"
 ▶ value: X {s: 1, e: 0, c: Array(1)}
 ▶ __proto__: Object
```

字段	解释
blockHash	交易所在区块的哈希值
blockNumber	交易所在区块的块号
from	交易发起者的地址
gas	交易发起者提供的gas
gasPrice	交易发起者配置的gas价格
hash	交易的hash值
input	表示是一个创建或调用智能合约交易
nonce	交易发起者在之前进行过的交易数量
r	交易签名
s	交易签名
to	交易接收者的地址
transactionIndex	交易在区块中的序号
v	交易签名
value	交易附带的货币量