以太坊私有链相关信息

参考：
https://github.com/xiaoping378/blog/blob/master/posts/
https://g2ex.github.io/2017/09/12/ethereum-guidance/

一，安装go-ethereum(geth)
它就包含区块链节点代码
1.ppa安装
2.源码下载编译安装
二，安装solidity编译器
apt install solc

三，搭建私有链
1.初始化genesis.json
geth --datadir data0 init genesis.json
geth --datadir data1 init genesis.json
2，启动私有链
geth --identity "TestNode" --rpc --rpcport "8545" --datadir data0 --port "30303" --nodiscover console
geth --identity "TestNode1" --rpc --rpcport "8546" --datadir data1 --port "30304" --nodiscover console
—identity：指定节点 ID；
—rpc：表示开启 HTTP-RPC 服务；
—rpcport：指定 HTTP-RPC 服务监听端口号（默认为 8545）；
—datadir：指定区块链数据的存储位置；
—port：指定和其他节点连接所用的端口号（默认为 30303）；
—nodiscover：关闭节点发现机制，防止加入有同样初始配置的陌生节点。

3，控制台操作(看看有没有帮助文档之类的东西)
//创建账户：
TestNode:"0xd0af90cad0df03d6a991c6304bf5cf2f88214a62"
TestNode1:"0xf5b0bb74ba6dcf657603c5e23283aa50bb98f010"

//解锁账户
personal.unlockAccount("0x77e80b72d8e58fd0f95baa05fa3b4fe9723ba3df")

//列出所有账户
eth.accounts

//查看余额

eth.getBalance(”address”)

//连接节点(必须chainId一样的才可以相互连接，注意端口号不要冲突)
1.admin.nodeInfo.enode
2.admin.addPeer(”enode:”)
3.admin.peers//查看连接到的其他节点信息 net.peerCount//查看连接的节点数量
4.节点连接之后可以相互同步块信息，但是各自创建的账户信息不能用eth.accounts查看到
5.也可以使用--bootnodes在启动时连接到其他节点

//转账交易
personal.unlockAccount(eth.accounts[0])
eth.sendTransaction({from:eth.accounts[0],to:”0xf5b0bb74ba6dcf657603c5e23283aa50bb98f010”,value:amount})

//查看交易池中待确认交易
txpool.status

//出一个块
miner.start(1);admin.sleepBlocks(1);miner.stop();

----------智能合约----------
1.编写sol合约文件
2.solc编译生成EVM二进制码
======= testContract.sol:TestContract =======
Binary:
60806040523480156100105760080fd5b5060c58061001f6000396000f3006080604052600436106043610603f576000357c0100000000000000000000000000000000000000000000000000000000900463ffffffff168063165c4a16146044575b600080fd5b348015604f57600080fd5b506076600480360381019080803590602001909291908035906020019092919050505050608c565b60405180828152602001915050604051809103906f35b60008183029050929150505600a165627a7a723058205625133d637e80fdf7c855bfb75c4e18737a7e1be3f2cd35c95225b3c8ecdbb00029
3.生成JSON ABI(Application Binary Interface)
======= testContract.sol:TestContract =======
Contract JSON ABI
[{”constant”:false,”inputs”:[{”name”:”a”,”type”:”uint256”},{”name”:”b”,”type”:”uint256”}],”name”:”multiply”,”outputs”:[{”name”:””,”type”:”uint256”}],”payable”:false,”stateMutability”:”nonpayable”,”type”:”function”}]
4.部署智能合约
//用变量 code 和 abi 记录上面两个值，注意在code前加上0x前缀
> myContract = eth.contract(abi) //生成合约对象
{
  abi: [{
      constant: false,
      inputs: [{...}, {...}],
      name: ”multiply”,

```
    outputs: [{...}],
    payable: false,
    stateMutability: "nonpayable",
    type: "function"
}],
eth: {
  accounts: ["0xd0af90cad0df03d6a991c6304bf5cf2f88214a62"],
  blockNumber: 20,
  coinbase: "0xd0af90cad0df03d6a991c6304bf5cf2f88214a62",
  compile: {
    lll: function(),
    serpent: function(),
    solidity: function()
  },
  defaultAccount: undefined,
  defaultBlock: "latest",
  gasPrice: 18000000000,
  hashrate: 0,
  mining: false,
  pendingTransactions: [],
  protocolVersion: "0x3f",
  syncing: false,
  call: function(),
  contract: function(abi),
  estimateGas: function(),
  filter: function(options, callback, filterCreationErrorCallback),
  getAccounts: function(callback),
  getBalance: function(),
  getBlock: function(),
  getBlockNumber: function(callback),
  getBlockTransactionCount: function(),
  getBlockUncleCount: function(),
  getCode: function(),
  getCoinbase: function(callback),
  getCompilers: function(),
  getGasPrice: function(callback),
  getHashrate: function(callback),
  getMining: function(callback),
  getPendingTransactions: function(callback),
  getProtocolVersion: function(callback),
  getRawTransaction: function(),
  getRawTransactionFromBlock: function(),
  getStorageAt: function(),
  getSyncing: function(callback),
  getTransaction: function(),
  getTransactionCount: function(),
  getTransactionFromBlock: function(),
  getTransactionReceipt: function(),
  getUncle: function(),
```

```
    getWork: function(),
    iban: function(iban),
    icapNamereg: function(),
    isSyncing: function(callback),
    namereg: function(),
    resend: function(),
    sendIBANTransaction: function(),
    sendRawTransaction: function(),
    sendTransaction: function(),
    sign: function(),
    signTransaction: function(),
    submitTransaction: function(),
    submitWork: function()
  },
  at: function(address, callback),
  getData: function(),
  new: function()
}

> contract = myContract.new({from:eth.accounts[0],data:code,gas:1000000})//发送部署的合约
INFO [05-24|14:34:24] Submitted contract creation fullhash=0xf932abdf8cc19dbeff352981e916598fd1f36354a70999849aeea1f22badb516 contract=0x81A02EA560A864DfC8839f0b9D0d05595BcbA472
{
  abi: [{
      constant: false,
      inputs: [{...}, {...}],
      name: "multiply",
      outputs: [{...}],
      payable: false,
      stateMutability: "nonpayable",
      type: "function"
  }],
  address: undefined,
  transactionHash: "0xf932abdf8cc19dbeff352981e916598fd1f36354a70999849aeea1f22badb516"
}

5.调用合约(因为没有contract上下文，如何在其他节点调用该合约呢)
contract.multiply.sendTransaction(2, 4, {from:eth.accounts[0]})



----------------------------------------------mist连接到私有链----------
----------------------------------------------
//钱包连接到上面的data0节点
./mist --rpc ../ethereumchain/data0/geth.ipc --node-datadir ../ethereumchain/data0/ --node-networkid 1025
```

再次启动时，需要挖一次矿才能正常同步(推断这也是一个交易行为导致的)

----------mist功能点----------

一，创建账户

二，创建钱包
1.合约地址
单签名钱包合约：0xF3943254F9E0Bd3445F5572c9715855c4F02C75f
创建多签钱包合约：0xeC67B1272d49716a9d6843D40D435C17403bAadb
钱包合约地址=导入钱包的地址
2.当新建多签钱包后，钱包的资金是为0的，哪怕某个账户里面是有余额的；多签钱包可以理解为以多人的名义在银行申请了一张空白卡片，这个卡片和某个账户已有卡片没什么关系。

三，转账
1.转账对象可以是账户，也可以是合约地址(如钱包合约地址)
2.转账花销：手续费+gas*gas price

四，合约
另外开一章

五,展示功能
1.交易历史
交易hash，交易时间，金额，from to，手续费，gas，gas price，所在区块及区块hash
2.账户余额

六，钱包备份
根据上面的操作配置，需要备份的文件是在数据目录(如data0)下的keystore。

----------mist合约功能----------

地址筛选：
//资源概括
https://blog.csdn.net/sinat_34070003/article/details/79126736
//系列文章
https://segmentfault.com/a/1190000014259779
https://ethfans.org/posts/101-noob-intro
//个人blog
http://liyuechun.org/page/6/#blog
//remix
https://ethfans.org/posts/deploying-smart-contract-with-remix

一，开发工具
//IDE工具集合
https://solidity-cn.readthedocs.io/zh/develop/
开发——>Open Remix IDE

二，合约源码实例(合约)
１．建立代币
２．启动众筹
３．创建区块链组织

三，部署合约
1.合约——>部署合约——>将sol源码拷贝到mist——>即时编译sol文件——>部署
2.部署成功——>挖矿——>进入合约交互页面——>使用合约中定义的函数等等
3.当我们调用increment时，是写入数据，需要花费gas，而getCount是从区块链读取数据，无需话费gas

注：
1.编写合约用Atom
2.部署合约可以用get命令行或者mist(可以先用命令将原理摸清楚)