如何搭建以太坊私链以及部署智能合约

本文是对前期搭建以太坊私链的一个总结，对搭建过程中出现的错误进行记录，希望能帮助到你们。

之所以采用 ubuntu 搭建以太坊私链，是因为以太坊对于 Linux 的支持性较好，安装也较为简单。我这里采用 的是 Ubuntu 14.04 系统。

一、安装 geth 客户端

打开 Ubuntu 命令行，输入以下命令：

```
sudo apt-get update

sudo apt-get installsoftware-properties-common

sudo add-apt-repository -yppa:ethereum/ethereum

sudo add-apt-repository -yppa:ethereum/ethereum-dev

sudo apt-get update

sudo apt-get install ethereum
```

这样就完成了以太坊客户端。

此时在命令行中输入 geth 命令，出现 geth 启动的信息则表示安装成功（此时链接的是共有链），或者输入 geth –help 出现命令行各种参数提示信息。

二、安装 solc

2.1 安装 solc 之前请确保安装过 npm,因为我们需要用 npm 安装 solc,而 npm 是包含在 nodejs 中，请先前往网下载 nodejs.

2.2 npm install -g solc

2.3 在命令行下输入 solc –hlep,出现一些指令使用则表面安装成功。

三、配置私链节点。

3.1 创世快设置

新建文件夹，命名随意，在此文件夹下创建 genesis.json 文件和 data 文件夹

genesis.json 的内容如下：

```
{

"config": {

 "chainId": 123456,

 "homesteadBlock": 0,

 "eip155Block": 0,

 "eip158Block": 0

},
```

```
"nonce":"0x0000000000000042",

"difficulty": "0x020000",

"mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000",

"coinbase": "0x0000000000000000000000000000000000000000",

"timestamp":"0x00",

"parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",

"extraData": "0x11bbe8db4e347b4e8c937c1c8370e4b5ed33adb3db69cbdb7a38e1e50b1b82fa",

"gasLimit": "0x4c4b40",

"alloc": {}

}
```


3.2 初始化

在命令行下进入刚才创建的文件夹，输入如下命令：
geth --datadir data init genesis.json

各参数代表的含义如下：

- init 表示初始化区块，后面跟着创世块的配置文件 genesis.json
- datadir 数据存放的位置

3.3 启动节点

geth --datadir data --networkid 123456 --rpc --rpccorsdomain "*" --nodiscover console

各参数代表的含义如下：

- networkid 设置当前区块链的网络 ID，用于区分不同的网络，1 表示公链
- rpc 表示启动 rpc 通信，可以进行智能合约的部署和调试
- console 表示启动命令行模式，可以在 Geth 中执行命令
- 执行成功后将进入区块链的 JavaScript 控制台环境

3.4 Geth JavaScript 控制台环境使用命令

- 创建新账号
  personal.newAccount()
  或者 personal.newAccount("123456")
- 查看节点信息
  admin.nodeInfo
- 挖矿
  开始挖矿 miner.start(1)
  停止挖矿 miner.stop()
- 查看当前矿工账号
  eth.coinbase 默认为第一个账户
- 修改矿工账号
  miner.setEtherbase(eth.accounts[1])
- 查看账户信息
  eth.accounts[0]
- 查看账户余额
  eth.getBalance(eth.accounts[0])
  或者 web3.fromWei(eth.getBalance(eth.accounts[0]), "ether")
- 解锁账号
  personal.unlockAccount(eth.accounts[0])
  使用账户资金前都需要先解锁账号

- 转账
  `eth.sendTransaction({from:eth.accounts[0],to:"0x587e57a516730381958f86703b1f8e970ff445d9", value:web3.toWei(3,"ether")})`
  使用 `txpool.status` 可以看到交易状态

- 查看区块数据
  `eth.blockNumber`
  `eth.getTransaction("0x0c59f431068937cbe9e230483bc79f59bd7146edc8ff5ec37fea6710adcab825")`
  `eth.getBlock(1)` 通过区块号查看区块

四.智能合约

4.1 编辑合约代码

创建一个 Token.sol 文件,内容如下：

```
contract Token {

 address issuer;

mapping (address => uint) balances;

 event Issue(address account,uint amount);

 eventTransfer(address from, address to, uint amount);

 function Token() {

  issuer = msg.sender;

 }



 function issue(address account, uint amount) {
```

```
if (msg.sender != issuer) throw;

balances[account] += amount;

}

functiontransfer(address to, uint amount) {

if (balances[msg.sender] < amount) throw;

balances[msg.sender] -= amount;

balances[to] += amount;



Transfer(msg.sender, to, amount);

}



function getBalance(address account) constant returns (uint) {

return balances[account];

}

}
```

这份代码实现了一个简单的 Token 合约功能。
issue 函数可以向充值以太到合约账户
transfer 函数可以向其他账号发送 token
getBalance 函数可以获取某个账号的 token 余额

4.2 编译与部署

- 压缩合约代码
  命令行下执行 cat Token.sol | tr '\n' ' '
  这条命令将代码中的换行符替换成空格，这样我们的代码就只有一行了。命令执行成功后将回显复制下来。

- 将合约代码保存为一个变量
  回到 Geth JavaScript 控制台，执行如下命令，等于号后面的内容就是我们刚才复制下来的压缩后的合约代码。

- var tokenSource = 'contract Token { address issuer; mapping (address => uint) balances; event Issue(address account, uint amount); event Transfer(address from, address to, uint amount); function Token() { issuer = msg.sender; } function issue(address account, uint amount) { if (msg.sender != issuer) throw; balances[account] += amount; } function transfer(address to, uint amount) { if (balances[msg.sender] < amount) throw; balances[msg.sender] -= amount; balances[to] += amount; Transfer(msg.sender, to, amount); } function getBalance(address account) constant returns (uint) { return balances[account]; } }';

- 编译
  var tokenCompiled = eth.compile.solidity(tokenSource);

- 若不成功，请参考 https://ethereum.stackexchange.com/questions/15435/how-to-compile-solidity-contracts-with-geth-v1-6 提供的替代方案

- 查看二进制代码
  tokenCompiled['<stdin>:Token'].code

- 查看 ABI
  tokenCompiled['<stdin>:Token'].info.abiDefinition

- 创建合约对象
  var contract = eth.contract(tokenCompiled['<stdin>:Token'].info.abiDefinition);

  var initializer = {from: web3.eth.accounts[0], data: tokenCompiled['<stdin>:Token'].code, gas: 300000};

  var token = contract.new(initializer)

输入命令 token 可以看到此时的 token 有 transactionHash 但是没有 address
执行 miner.start(1) 一段时间后停止，我们的合约就发布到了链上

4.3 与合约进行交互

- 充值

  ```
  personal.unlockAccount(eth.accounts[0])
  token.issue.sendTransaction(eth.accounts[0], 100, {from: eth.accounts[0]});
  miner.start(1)
  miner.stop()
  ```

- 发送 token

  ```
  token.transfer(eth.accounts[1], 30, {from: eth.accounts[0]})
  miner.start(1)
  miner.stop()
  ```

- 查看余额

  ```
  token.getBalance()
  ```