以太坊本地私有链开发环境搭建！通过本文所述方法和项目中的脚本，我们可以快速的搭建好自己的私链进行开发测试。

仓库中包含的工具有：

一个测试账户导入脚本，在首次部署时将五个测试账户私钥导入以太坊节点。

一个genesis.json配置文件，为对应的五个测试账户提供初始资金(以太币)，方便开发测试。

一个快速启动私有链节点并进入交互模式的脚本。

一个合约样例：contracts/Token.sol。这是一个使用合约语言Solidity编写的智能合约。Token合约的功能是发行一种token(可以理解为货币，积分等等)，只有合约的创建者有发行权，token的拥有者有使用权，并且可以自由转账。

测试账户私钥是放在Github上的公开数据，千万不要用于正式环境中或者公有链上。如果在测试环境之外的地方使用这些私钥，你的资金将会被窃取!

准备

在本地安装好go-ethereum和solc,
可以执行geth和solc命令。如果操作系统是ubuntu,
安装官方的ethereum安装包即可。

将本仓库通过git clone命令下载到本地。

安装expect，工具脚本用它来自动化一些过程。例如在ubuntu上:sudo apt-get install expect

启动geth

进入本仓库目录:cd ethereum-bootstrap

导入测试账户私钥:./bin/import_keys.sh

启动私有链节点:./bin/private_blockchain.sh. 启动成功后可以看到类似如下输出:

此时以太坊交互式控制台已经启动，我们可以开始测试和开发了。

注意：工具脚本假设你的geth安装在默认位置, 可以直接通过geth执行。如果geth命令安装在非标准的位置，可以设置GETH环境变量指定geth可执行文件的路径。例如:

GETH=/some/weird/dir/geth ./bin/import_keys.sh

使用以太坊控制台编译和部署智能合约

在contracts目录下有一个智能合约样例文件Token.sol,
通过Solidity语言实现了基本的代币功能, 合约持有者可以发行代币,
使用者可以互相转账.

我们可以使用以太坊控制台来编译部署这个合约.以太坊控制台是最基本的工具，使用会比较繁琐.社区也提供了其他更加方便的部署工具，此处不做讨论.

第一步，我们先把合约代码压缩为一行.新建一个ssh session,
切换到geth用户环境su - geth, 然后输入：cat contracts/Token.sol | tr 'n' ' '.

切换到以太坊控制台，把合约代码保存为一个变量:

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]

然后编译合约代码：

var tokenCompiled = web3.eth.compile.solidity(tokenSource);

通过tokenCompiled.Token.code可以看到编译好的二进制代码，通过tokenCompiled.Token.info.abiDefinition可以看到合约的ABI.

接下来我们要把编译好的合约部署到网络上去.

首先我们用ABI来创建一个javascript环境中的合约对象：

```
var contract = web3.eth.contract(tokenCompiled.Token.info.abiDefinition);
```

我们通过合约对象来部署合约：

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

var callback = function(e, contract){

if(!e) {

if(!contract.address) {

console.log("Contract transaction send: TransactionHash: " +
contract.transactionHash + " waiting to be mined...");

} else {

console.log("Contract mined!");

console.log(contract);

}

}

};

var token = contract.new(initializer, callback);
```

contract.new方法的第一个参数设置了这个新合约的创建者地址from,
这个新合约的代码data, 和用于创建新合约的费用gas.gas是一个估计值，只要比所需要的gas多就可以，合约创建完成后剩下的gas会退还给合约创建者.

contract.new方法的第二个参数设置了一个回调函数，可以告诉我们部署是否成功.

contract.new执行时会提示输入钱包密码.执行成功后，我们的合约Token就已经广播到网络上了.此时只要等待矿工把我们的合约打包保存到以太坊区块链上，部署就完成了.

在公有链上，矿工打包平均需要15秒，在私有链上，我们需要自己来做这件事情.首先开启挖矿：

miner.start(1)

此时需要等待一段时间，以太坊节点会生成挖矿必须的数据，这些数据都会放到内存里面.在数据生成好之后，挖矿就会开始，稍后就能在控制台输出中看到类似：

:hammer:Mined block

的信息，这说明挖到了一个块，合约已经部署到以太坊网络上了!此时我们可以把挖矿关闭：

miner.stop(1)

接下来我们就可以调用合约了.先通过token.address获得合约部署到的地址,以后新建合约对象时可以使用.这里我们直接使用原来的contract对象：

// 本地钱包的第一个地址所持有的token数量

>

token.getBalance(web3.eth.accounts[0])

0

// 发行100个token给本地钱包的第一个地址

>

token.issue.sendTransaction(web3.eth.accounts[0], 100. {from: web3.eth.accounts[0]});

I1221 11:48:30.512296 11155 xeth.go:1055] Tx(0xc0712460a826bfea67d58a30f584e4bebdbb6138e7e6bc1dbd6880d2fce3a8ef) to:

0x37dc85ae239ec39556ae7cc35a129698152afe3c

"0xc0712460a826bfea67d58a30f584e4bebdbb6138e7e6bc1dbd6880d2fce3a8ef"

// 发行token是一个transaction, 因此需要挖矿使之生效

>

miner.start(1)

:hammer:Mined block

>

miner.stop(1)

// 再次查询本地钱包第一个地址的token数量

>

token.getBalance(web3.eth.accounts[0])

100

// 从第一个地址转30个token给本地钱包的第二个地址

>

token.transfer.sendTransaction(web3.eth.accounts[1], 30. {from: web3.eth.accounts[0]})

I1221 11:53:31.852541 11155 xeth.go:1055] Tx(0x1d209cef921dea5592d8604ac0da680348987b131235943e372f8df35fd43d1b) to: 0x37dc85ae239ec39556ae7cc35a129698152afe3c

"0x1d209cef921dea5592d8604ac0da680348987b131235943e372f8df35fd43d1b"

>

miner.start(1)

>

miner.stop(2)

>

token.getBalance(web3.eth.accounts[0])

70

>

token.getBalance(web3.eth.accounts[1])

30

其他

私有链的所有数据都会放在仓库根目录下的data目录中，删除这个目录可以清除所有数据，重新启动新环境。

做完这些之后你应该对在以太坊私有链上进行开发有了一个大概的了解吧，如果还想学习更多知识，一可以看一看上面执行的脚本代码，到底干了些什么，用了哪些命令行参数，二可以阅读正在ethfans上更新的solidity文档中文版。