# 一、开发环境

### 开发语言

Solidity 是一门面向合约的、为实现智能合约而创建的高级编程语言。这门语言受到了 C++，Python 和 Javascript 语言的影响，设计的目的是能在以太坊虚拟机（EVM）上运行。

Solidity 是静态类型语言，支持继承、库和复杂的用户定义类型等特性。

使用 Solidity 语言，可以为投票、众筹、秘密竞价（盲拍）、多重签名的钱包以及其他应用创建合约。

（目前尝试 Solidity 编程的最好的方式是使用 Remix （需要时间加载，请耐心等待）。Remix 是一个基于 Web 的 IDE，它可以让你编写 Solidity 智能合约，然后部署并运行该智能合约。）

### 开发工具

1. 以太坊客户端（geth）（搭建以太坊私链）

客户端 语言 开发者

go-ethereum Go 以太坊基金会

Parity Rust Ethcore

cpp-ethereum C++ 以太坊基金会

pyethapp Python 以太坊基金会

ethereumjs-lib Javascript 以太坊基金会

Ethereum(J) Java <ether.camp>

ruby-ethereum Ruby Jan Xie

ethereumH Haskell BlockApps

1. Remix——智能合约在线IDE（编译solidity代码）

<https://remix.ethereum.org/>

以太坊官方推荐的智能合约开发IDE，适合新手，可以在浏览器中快速部署测试智能合约。

1. Node.js（连接前端与智能合约）

* Web3.js（与智能合约交互）

web3.js是以太坊提供的一个Javascript库，它封装了以太坊的JSON RPC API，提供了一系列与区块链交互的Javascript对象和函数，包括查看网络状态，查看本地账户、查看交易和区块、发送交易、编译/部署智能合约、调用智能合约等，其中最重要的就是与智能合约交互的API。

* Express.js（路由Http请求）

Express 是一个简洁而灵活的 node.js Web应用框架, 提供了一系列强大特性帮助你创建各种 Web 应用，和丰富的 HTTP 工具。

使用 Express 可以快速地搭建一个完整功能的网站。

Express 框架核心特性：

* 可以设置中间件来响应 HTTP 请求。
* 定义了路由表用于执行不同的 HTTP 请求动作。
* 可以通过向模板传递参数来动态渲染 HTML 页面。

# 二、以太坊私链搭建

### 下载安装Geth

<https://ethfans.org/wikis/Ethereum-Geth-Mirror>

### 下载安装Node.js（里面自带了npm）

<https://nodejs.org/en/download/>

使用npm安装Web3.js

npm install web3 -g

使用npm安装Express.js

npm install express -g

### 建立创世区块

* 新建创世区块文件genesis.json，并填入以下代码

{

    "config": {

        "chainId": 15,

        "homesteadBlock": 0,

        "eip155Block": 0,

        "eip158Block": 0

    },

    "coinbase" : "0x0000000000000000000000000000000000000000",

    "difficulty" : "0x40000",

    "extraData" : "",

    "gasLimit" : "0xffffffff",

    "nonce" : "0x0000000000000042",

    "mixhash" : "0x0000000000000000000000000000000000000000000000000000000000000000",

    "parentHash" : "0x0000000000000000000000000000000000000000000000000000000000000000",

    "timestamp" : "0x00",

    "alloc": { }

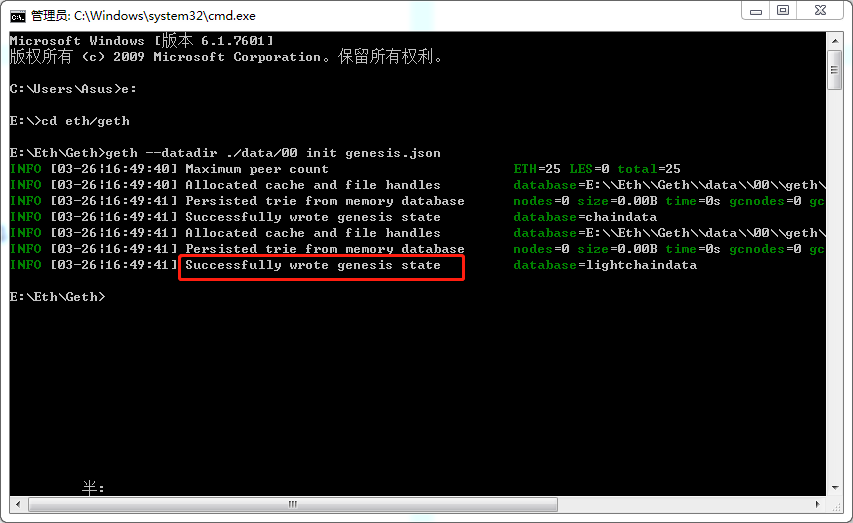
}

解释一下各个参数的作用：

|  |  |
| --- | --- |
| **mixhash** | 与nonce配合用于挖矿，由上一个区块的一部分生成的hash。注意他和nonce的设置需要满足以太坊的Yellow paper, 4.3.4. Block Header Validity, (44)章节所描述的条件。. |
| **nonce** | nonce就是一个64位随机数，用于挖矿，注意他和mixhash的设置需要满足以太坊的Yellow paper, 4.3.4. Block Header Validity, (44)章节所描述的条件。 |
| **difficulty** | 设置当前区块的难度，如果难度过大，cpu挖矿就很难，这里设置较小难度 |
| **alloc** | 用来预置账号以及账号的以太币数量，因为私有链挖矿比较容易，所以我们不需要预置有币的账号，需要的时候自己创建即可以。 |
| **coinbase** | 矿工的账号，随便填 |
| **timestamp** | 设置创世块的时间戳 |
| **parentHash** | 上一个区块的hash值，因为是创世块，所以这个值是0 |
| **extraData** | 附加信息，随便填，可以填你的个性信息 |
| **gasLimit** | 该值设置对GAS的消耗总量限制，用来限制区块能包含的交易信息总和，因为我们是私有链，所以填最大。 |

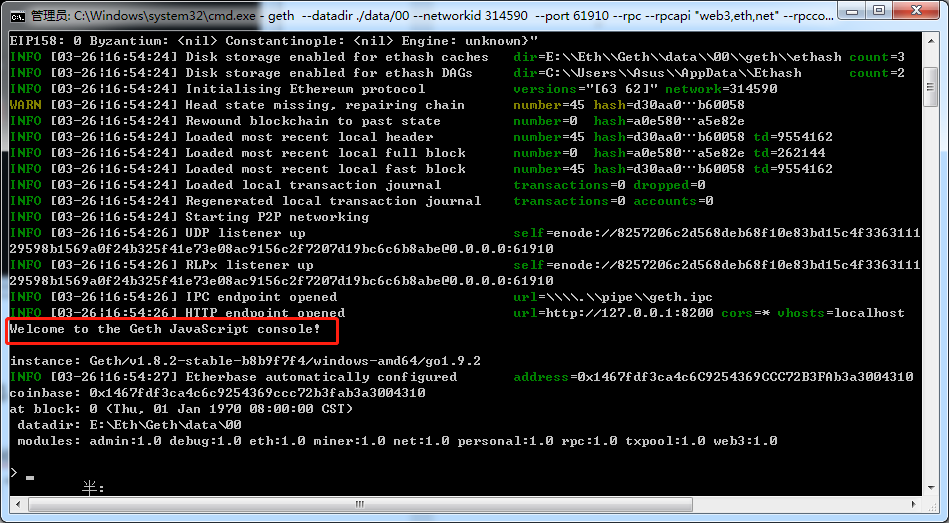
* 初始化创世节点，并设置data目录

geth --datadir ./data/00 init genesis.json



出现红框中提示，证明初始化完毕！

### 启动私有链节点，并设置端口

geth --datadir E:\\eth\\Geth\\data\\00 --networkid 20180319 --rpc --rpccorsdomain "\*" --nodiscover --port 16333 --rpcapi 'web3,eth,net,personal' --rpcport 8200 console

出现红框中提示，证明节点启动成功！

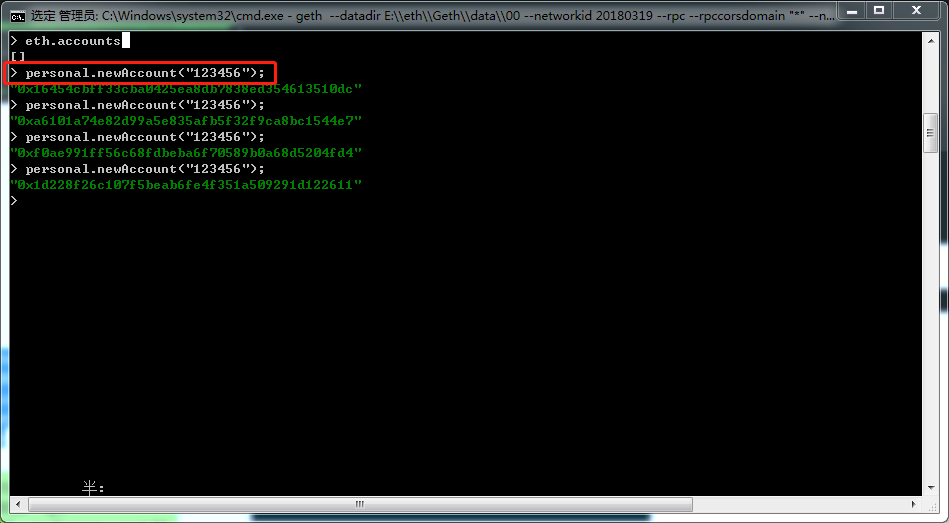
 以太坊客户端Geth命令用法-参数详解：[www.cnblogs.com/tinyxiong/p/7918706.html](http://www.cnblogs.com/tinyxiong/p/7918706.html)

|  |  |
| --- | --- |
| **identity** | 区块链的标示，随便填写，用于标示目前网络的名字 |
| **init** | 指定创世块文件的位置，并创建初始块 |
| **datadir** | 设置当前区块链网络数据存放的位置 |
| **port** | 网络监听端口 |
| **rpc** | 启动rpc通信，可以进行智能合约的部署和调试 |
| **rpcapi** | 设置允许连接的rpc的客户端，一般为db,eth,net,web3 |
| **networkid** | 设置当前区块链的网络ID，用于区分不同的网络，是一个数字 |
| **console** | 启动命令行模式，可以在Geth中执行命令 |

为了构建私有链 ，需要在Geth启动时加入一些参数，Geth参数含义如下：

### 创建测试账户

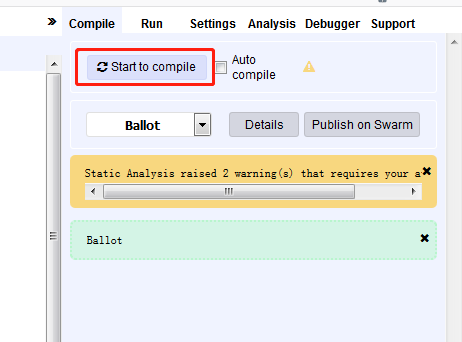
personal.newAccount("123456");



### 使用Remix，开发和部署智能合约

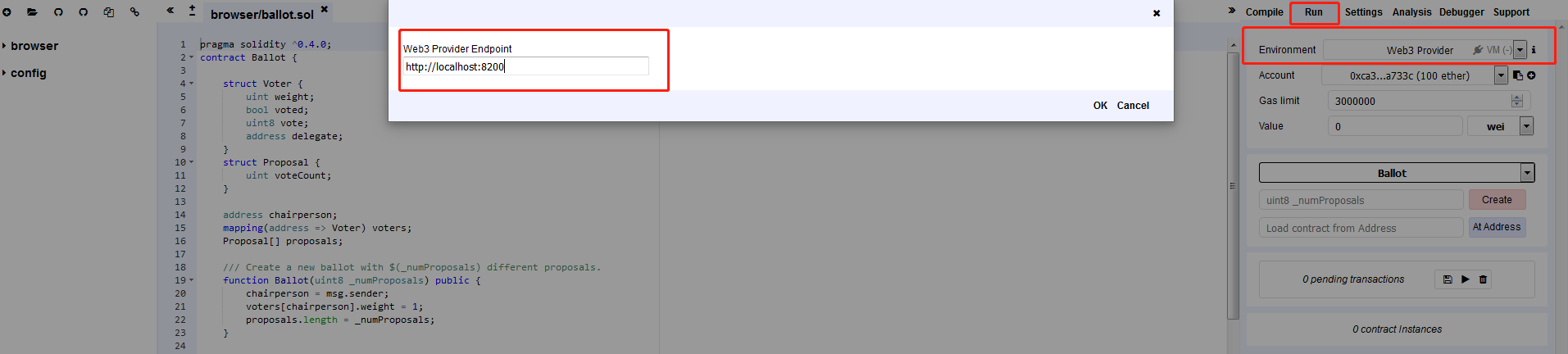
在remix中编写智能合约

合约代码完成后，开始编译

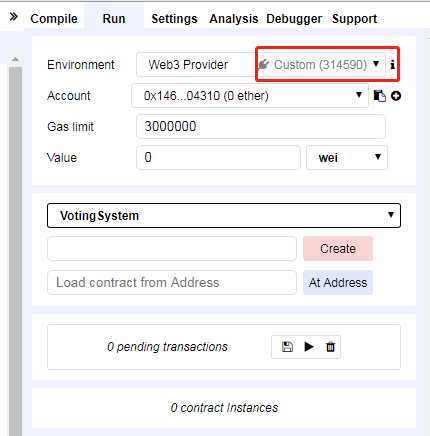


使用Web3 provider连接以太坊私链

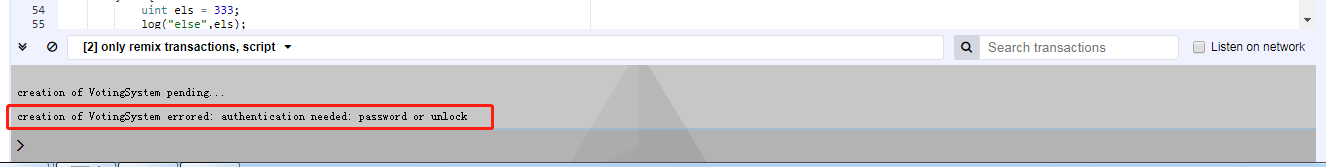
Run -> Environment 下拉选择 Web3 provider，在弹出的提示框中，输入之前启动节点定义的端口（--rpcport 8200我这里定义的是8200）



出现红框中提示，证明连接成功！

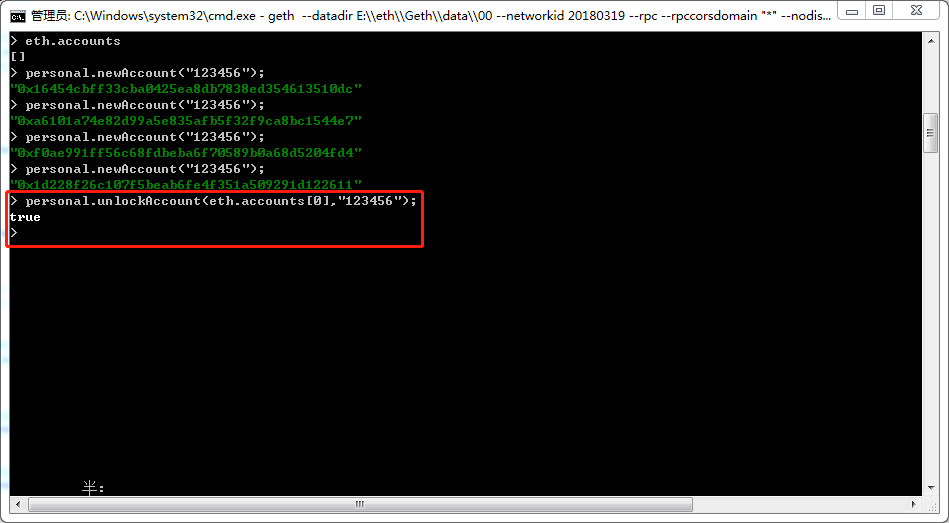


再点击下面的Create按钮，remix控制台提示解锁账户



在geth中解锁账户（参数为解锁账户，密码）

personal.unlockAccount(eth.accounts[0],"123456");



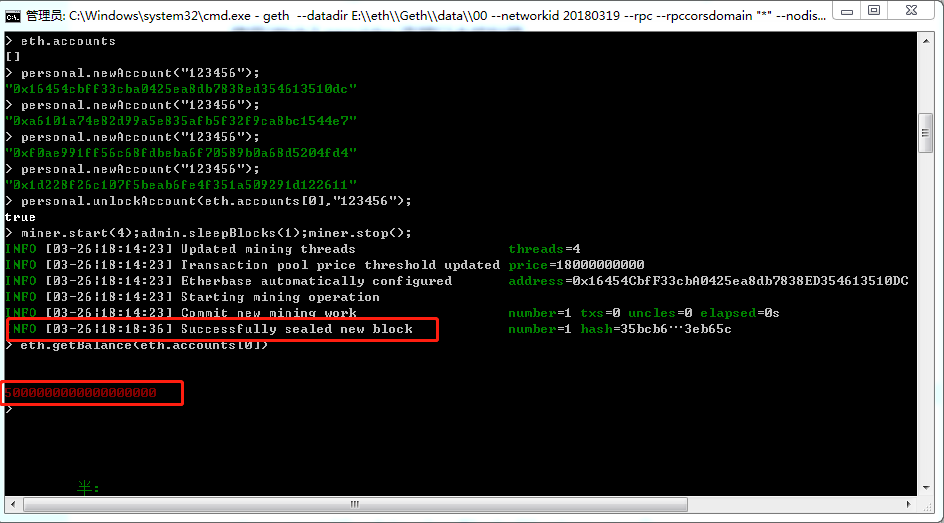
回到remix，再次Create，如果部署合约所用账户余额不足，会提示需要以太币，否则跳过以下挖矿步骤



miner.start(4);admin.sleepBlocks(1);miner.stop();

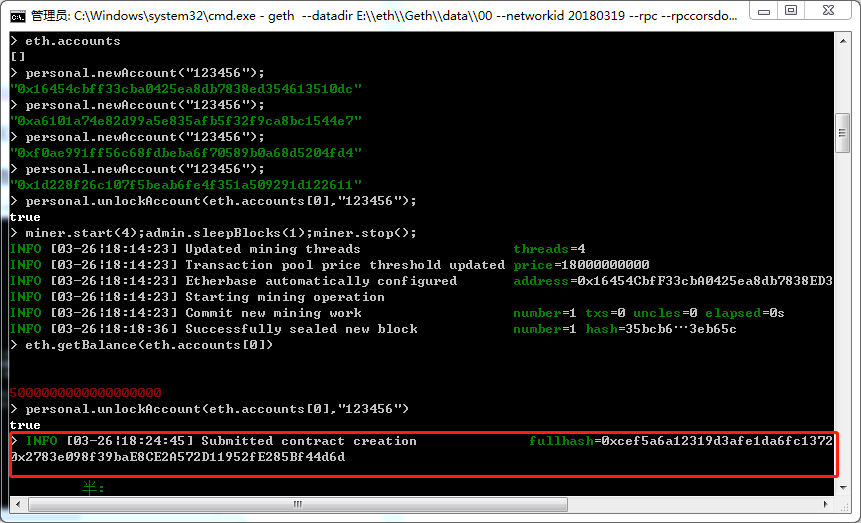
方法解释：开启4个线程挖矿，挖到一个区块就停止

再执行eth.getBalance(eth.accounts[0])查询余额



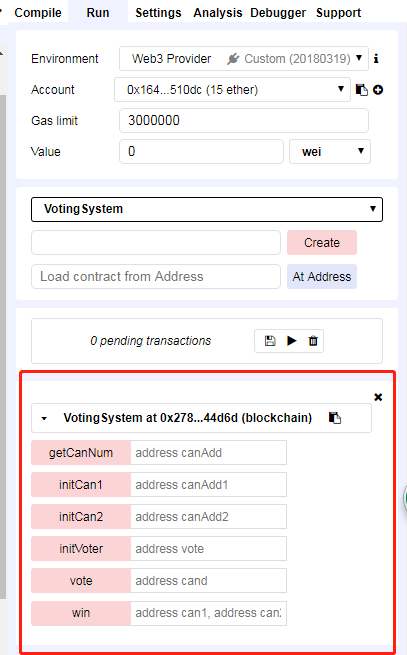
先在Geth中解锁账户

再次回到remix点击Create，提示如下

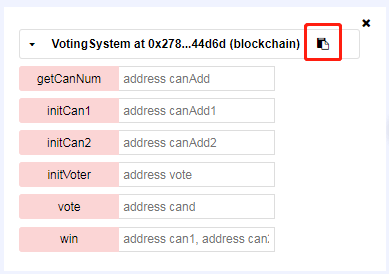


再执行挖矿（miner.start(4);admin.sleepBlocks(1);miner.stop();

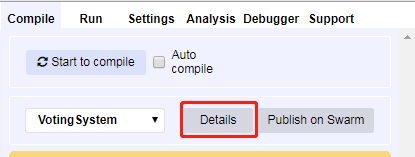
），会多出红框中的信息，到此部署成功



点击复制按钮，复制合约地址（web.js中需要使用）



在remix的compile菜单中，点击下面details进入，在WEB3DEPLOY中查看合约abi（web3.eth.contract的中括号里的信息）（web.js中需要使用）





### 在js中，使用Web3操作合约对象

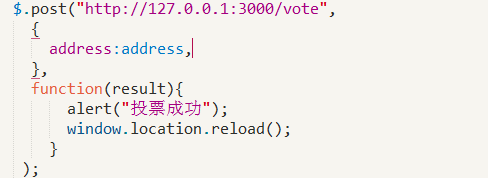


### 设置ajax跨域



### 前后端交互。即html和nodejs的ajax交互，这里用的是同步请求。

Html：在这里我们自己编写了一个简单的投票页面。这里展示ajax-post请求的重要的代码：



Nodejs：

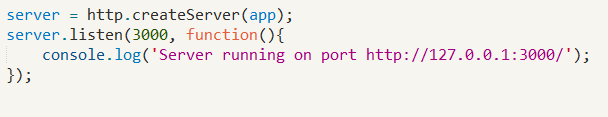


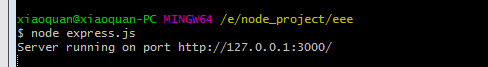
获取票数：myContract.getCanNum.call；这里的call方法只读取不写入。

投票：myContract.vote.sendTransaction；这里的sendTransaction执行程序并写入区块中。

web3.personal.unlockAccount("0xd9ee24af31eacff4e8f07b260152a86d043b1c95", "123456"); 解锁账户，可以在后台解锁。

### 监听端口





**这里是最终展示的页面效果：**

