简单投票DAPP

接下来我们要开始动手真正的做一个DApp，尽管它是很简单的一个投票应用，但会包含完整的工作流程和交互页面。构建这个应用的主要步骤如下：

1. 基础知识了解
2. 开发环境配置
3. 编写合约并在线测试
4. 部署合约
5. 网页交互

## 基本知识了解

Dapp(Decentralized application，去中心化应用)：运行在分布式网络上，参与者的信息被安全保护（也可能是匿名的），通过网络节点不同人，进行去中心化操作的应用。

Nodejs：基于 Chrome V8 引擎的 JavaScript 运行环境，通过JavaScript语言开发web服务端的东西。有非阻塞，事件驱动I/O等特性。传统的服务器（比如Apache）每次一个新用户连到你的网站上，你的服务器就得开一个连接,非常占资源。而nodejs就可以解决这个问题，非常适合分布式设备上运行数据密集型应用。

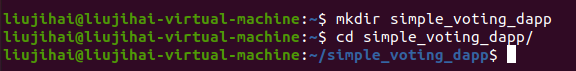
ganache-cli:模拟区块链，相对geth更加方便的测试平台。具有一键创建一个区块链，快速打包确认交易，直接提供多个测试账户等特点。

Web3：以太坊区块链交互的javascript库，应用和节点进行通信的中间层。Web3.js有0.2x

Solc: 合约代码的一个编译器

## 开发环境配置

新建一个simple\_voting\_dapp,并打开它

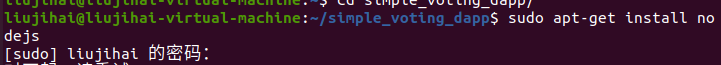


[小提示：如果你的ubuntu不能适应屏幕，视觉上不舒服，可以输入 sudo apt-get install -y open-vm\*]



Nodejs安装

1. 安装node.js sudo apt-get install nodejs



查看nodejs版本 node -v



下载npm sudo apt install npm



查看版本 npm -v



将nodejs更新到最新版本(不然只能下载web3 1.X版本)

Sudo npm install n -g



Sudo n stable

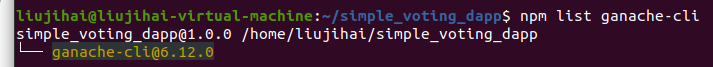




1. Ganache-cli安装 npm install ganache-cli



查看版本 npm list ganache-cli



1. Solc安装，智能合约的版本是什么就下载什么版本

Npm install solc@0.4.22



查看版本 npm list solc



1. Web3.js下载。需现下载git，不然又只能下载web3 的1.x版本

Sudo apt-get install git



Npm install web3@^0.20.0



好的，现在我们的环境就此配置完成！！！

## 编写合约Remix在线测试

1.Solidity合约

我们会写一个叫做Voting的合约，这个合约并不复杂，当然有很多地方可以改进，但是我们主要是熟悉开发的流程，如果同学们想要更加完善的合约代码，请自行努力吧。

这个合约有以下内容：

一个构造函数，用来初始化一些候选者。

一个用来投票的方法(对投票数加1)。voteForCandidate()

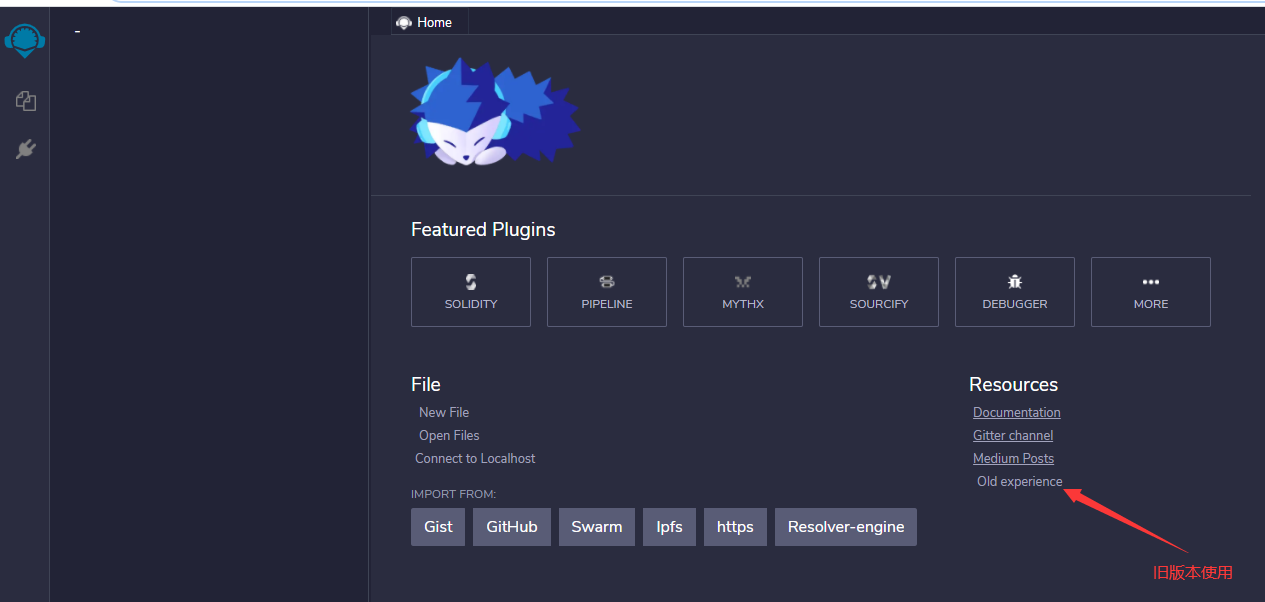
一个返回候选者所获得的总票数的方法。totalVotesFo()

一个用来检测是否为有效的投票地址：validCandidat()

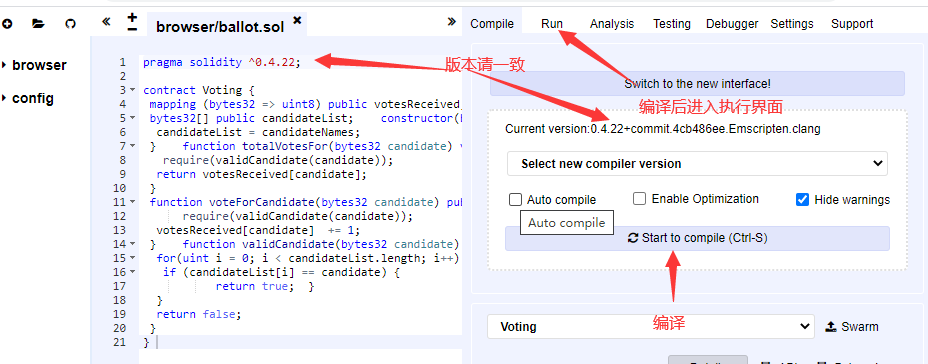
|  |  |
| --- | --- |
| 1.pragma solidity ^0.4.22;  2.contract Voting {  3. mapping (bytes32 => uint8) public votesReceived;  4.bytes32[] public candidateList;  5.constructor(bytes32[] candidateNames) public {  6. candidateList = candidateNames; }  7.function totalVotesFor(bytes32 candidate) view public returns (uint8) {  8.require(validCandidate(candidate));  9. return votesReceived[candidate]; }  10.function voteForCandidate(bytes32 candidate) public {  11. require(validCandidate(candidate));  12.votesReceived[candidate] += 1; }  13.function validCandidate(bytes32 candidate) view public returns (bool) {  14. for(uint i = 0; i < candidateList.length; i++) {  15. if (candidateList[i] == candidate) {  16. return true; } }  17. return false;  }  } | 1.指定代码将会哪个版本的编译器进行编译  3.mapping 相当于一个关联数组或者是字典，是一个键值对。mapping  votesReceived 的键是候选者的名字，类型为 bytes32  4.在solidity没有可直接获得候选者的key，只能单独管理一个候选者数组candidateNames  12.注意votesReceived[key]有一个默认值为0所有我们不需要初始化 |

2.接下来我们用solc的集成开发工具Remix进行部署和测试，成功后再部署到私有链中

Remix网址：http://remix.ethereum.org/



进入旧版本(旧版本比较直观)



进入执行阶段，构造函数deploy时输入构造函数的参数：

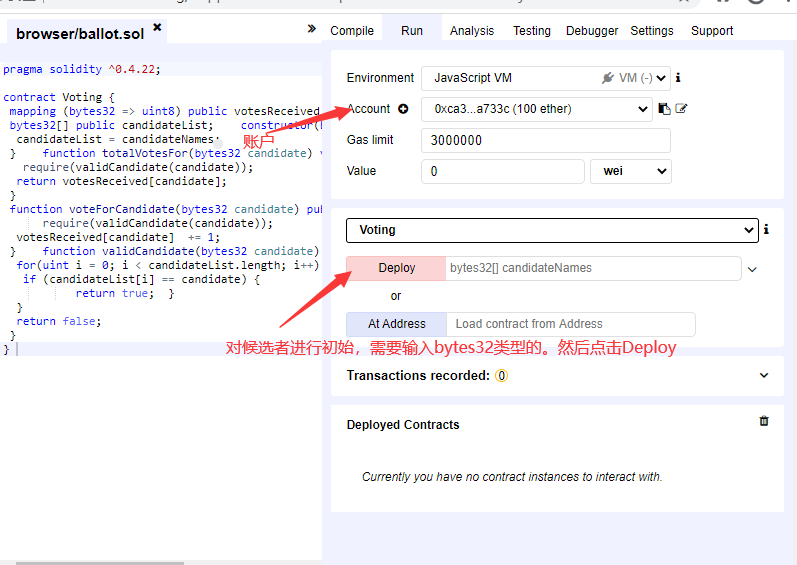
["0xd4967590eb024589dfb6b9e48a576eb49ebc19d764b0d1d67dc21975e7258e97",

"0x0000000000000000000000000000000000000000000000000000000000000001",

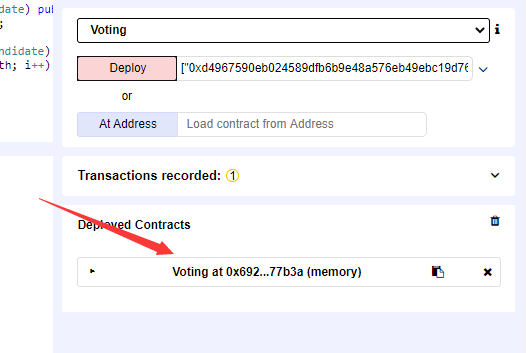
"0x0000000000000000000000000000000000000000000000000000000000000002",

"0x0000000000000000000000000000000000000000000000000000000000000003",

"0x065e0be95fb43db528a20ba65c0e575e33cd4a9e1ca089dba4efff24596e8553"]



点击deploy初始化后就会生成相应的候选者和投票的函数



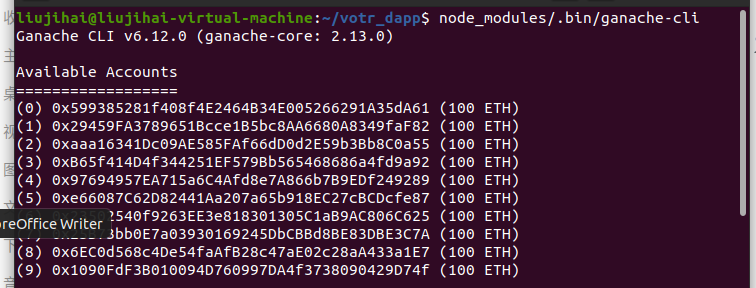
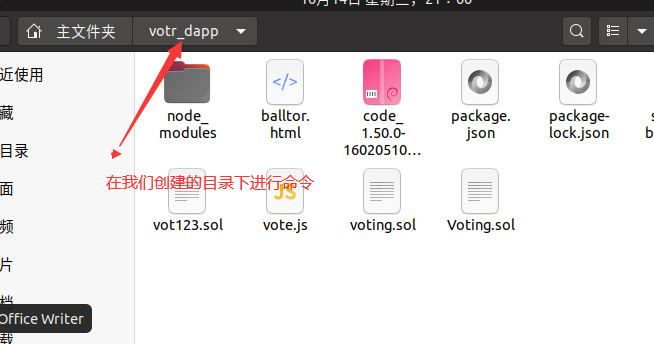


从现在开始我们的代码就正式的调试成功完成了。下面我们就开始在终端进行编译和部署到区块链中吧。

## 部署合约

在进行部署之前我们需先在一个终端窗口打开ganache-cli,确保模拟区块链的存在

node\_modules/.bin/ganache-cli

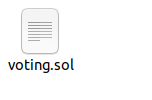


再打开一个终端进行部署

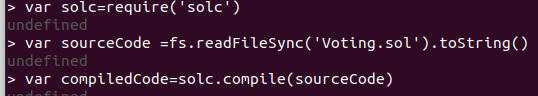
|  |  |
| --- | --- |
| 首先，在终端中运行 node 进入 node 控制台 | node |
| 看引入web3是否成功  初始化 web3 对象 | var Web3=require('web3')  var web3=new Web3(new Web3.providers.HttpProvider('http://localhost:8545')) |
| 看是否连接区块链 | web3.isConnected() |
| 向区块链查询获取所有的账户 | web3.eth.accounts |



将我们写好的合约代码放入我们最最开始所创建的simple\_voting\_dapp里，进行编译



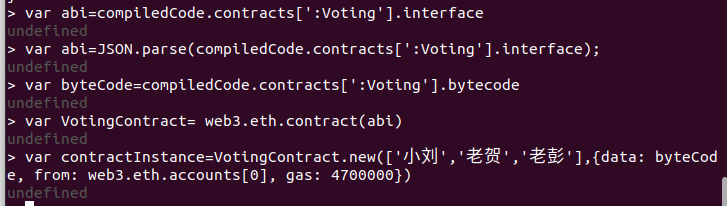
|  |  |
| --- | --- |
| 从 Voting.sol 中加载代码并绑定到一个 string 类型的变量 | var solc=require('solc')  var sourceCode =fs.readFileSync('vot123.sol').toString() |
| 成功地编译好合约给 compiledCode 对象 | var compiledCode=solc.compile(sourceCode) |
| 查看compiledCode你会发现俩个重要的字段abi和bytecode |  |
| compiledCode.contracts[':Voting'].bytecode | 这就是 Voting.sol 编译好后的字节码。也是要部署到区块链上的代码。 |
| compiledCode.contracts[':Voting'].interface | 这是一个合约的接口或者说模板（叫做 abi 定义），它告诉了用户在这个合约里有哪些方法。 |



现在将合约部署到区块链上。为此，你必须先通过传入 abi 定义来创建一个合约对象 VotingContract。然后用这个对象在链上部署并初始化合约。

|  |  |
| --- | --- |
| 将abi转换为json类型，因为web3.eth.contract()方法里是接受json类型 | var abi=compiledCode.contracts[':Voting'].interface  var abi=JSON.parse(compiledCode.contracts[':Voting'].interface); |
| 编译好的字节码，部署在区块链上的代码 | var byteCode=compiledCode.contracts[':Voting'].bytecode |
| 创建合约对象 | var VotingContract= web3.eth.contract(abi) |
| VotingContract.new 将合约部署到区块链 | var contractInstance=VotingContract.new(['小刘','老贺','老彭'],{data: byteCode, from: web3.eth.accounts[0], gas: 4700000}) |







我们已经部署了合约，并有了一个合约实例（变量 contractInstance），我们可以用这个实例与合约进行交互。

在区块链上有上千个合约。那么，如何识别你的合约已经上链了呢？答案是找到已部署合约的地址：ContractInstance.address. 当你需要跟合约进行交互时，就需要这个部署地址和我们之前谈到的 abi 定义.



## 网页交互

至此，大部分的工作都已完成，我们还需要做的事情就是创建一个简单的 html，里面有候选者姓名并调用投票命令（我们已经在Remix里试过）。你可以在右侧找到 html 代码和 js 代码。将它们放到 chapter1 目录，并在浏览器中打开 index.html。

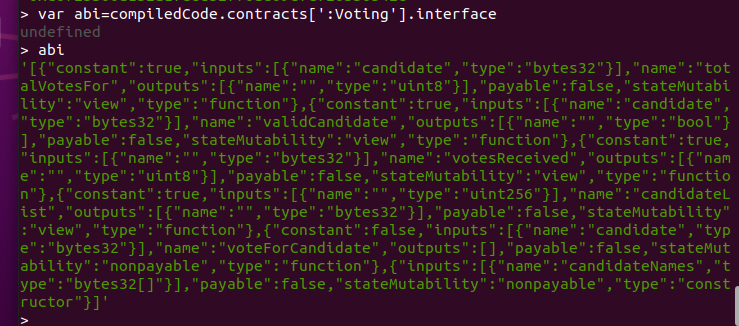
|  |
| --- |
| !DOCTYPE html>  <html>  <head>  <title>投票</title>  <link href='https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css' rel='stylesheet' type='text/css'>  </head>  <body class="container">  <h1 align="center">简单投票DAPP</h1>  <div class="table-responsive">  <table class="table table-bordered">  <thead>  <tr>  <th>候选者</th>  <th>票数</th>  </tr>  </thead>  <tbody>  <tr>  <td>小刘</td>  <td id="candidate-1"></td>  </tr>  <tr>  <td>老贺</td>  <td id="candidate-2"></td>  </tr>  <tr>  <td>老彭</td>  <td id="candidate-3"></td>  </tr>  </tbody>  </table>  </div>  <input type="text" id="candidate" />  <a href="#" onclick="voteForCandidate()" class="btn btn-primary">投票</a>  <div id="msg"></div>  </body>  <script src="https://cdn.jsdelivr.net/gh/ethereum/web3.js/dist/web3.min.js"></script>  <script src="http://libs.baidu.com/jquery/2.1.4/jquery.min.js"></script>  <script src="./index.js"></script>  </html> |

|  |
| --- |
| 1.<head>中用 link 形式引入 bootstrap 的 css 类型库，以下 container、table-responsive 等 class 均来自 bootstrap  2. <a>超链接形式的按键 btn，href=“#”为跳转至本页，即不跳转；onclick 指向js中方法  3. <script>引入了web3,js的在线文档。 |

Index.js

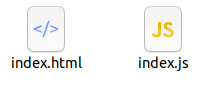
|  |
| --- |
| var web3 = new Web3(new Web3.providers.HttpProvider('http://127.0.0.1:8545'));  var abi = JSON.parse('[{"constant":true,"inputs":[{"name":"candidate","type":"bytes32"}],"name":"totalVotesFor","outputs":[{"name":"","type":"uint8"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":true,"inputs":[{"name":"candidate","type":"bytes32"}],"name":"validCandidate","outputs":[{"name":"","type":"bool"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":true,"inputs":[{"name":"","type":"bytes32"}],"name":"votesReceived","outputs":[{"name":"","type":"uint8"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":true,"inputs":[{"name":"","type":"uint256"}],"name":"candidateList","outputs":[{"name":"","type":"bytes32"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":false,"inputs":[{"name":"candidate","type":"bytes32"}],"name":"voteForCandidate","outputs":[],"payable":false,"stateMutability":"nonpayable","type":"function"},{"inputs":[{"name":"candidateNames","type":"bytes32[]"}],"payable":false,"stateMutability":"nonpayable","type":"constructor"}]');  var contractAddr = '0x149a4feb23f9705f40ea7a23e2e0d48eb079da4b';  var contractInstance = new web3.eth.Contract(abi,contractAddr);  var candidates = {'小刘':'candidate-1','老贺':'candidate-2','老彭':'candidate-3'};  var account;  web3.eth.getAccounts().then(function(accounts){  account = accounts[0];  });  function voteForCandidate(){  let candidateName = $("#candidate").val();  let candidateNameHex = web3.utils.asciiToHex(candidateName);  contractInstance.methods.voteForCandidate(candidateNameHex).send({from:account}).then(function(receipt){  $("#msg").html("已投给: "+ candidateName + "<br>交易哈希: " + receipt.transactionHash + "<br>投票人地址: " + account);  contractInstance.methods.totalVotesFor(candidateNameHex).call(function(err,res){  $('#'+candidates[candidateName]).html(res.toString());  });  });  }  $(document).ready(function(){  var candidateList = Object.keys(candidates);  for (let i = 0; i < candidateList.length; i++){  let name = candidateList[i];  let nameHex = web3.utils.asciiToHex(name);  let count = contractInstance.methods.totalVotesFor(nameHex).call(function(err,res){  $("#"+candidates[name]).html(res.toString());  });  }  }) |

红色标注：



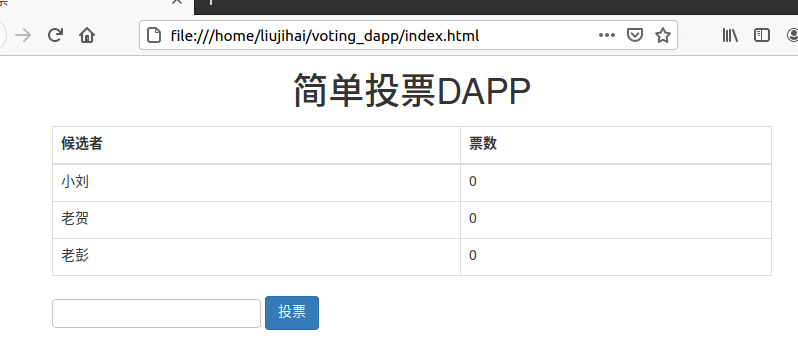
黄色标注：





为此我们的开发流程就全部完成了，让我们看一看界面，欣赏一下自己的成果吧！！

1.初始的候选者票数都为0



2.投票



再来让我们看看区块链都记录了哪些东西吧！！！

