

《区块链》期末 project 前期热身报告

一、以太坊的安装、私有链创世区块搭建、私有链节点的加入(选做)

1>以太坊的安装

1. ubuntu 下安装 geth 客户端

Geth 官方安装指南：

<https://github.com/ethereum/go-ethereum/wiki/Building-Ethereum>

具体命令行命令：







Installing from PPA

```
sudo apt-get install software-properties-common
sudo add-apt-repository -y ppa:ethereum/ethereum
sudo apt-get update
sudo apt-get install ethereum
```

2.安装图像化客户端 Mist

官方下载地址：<https://github.com/ethereum/mist/releases/>

下载选择：

 Ethereum-Wallet-installer-0-11-1.exe	127 MB
 Ethereum-Wallet-linux32-0-11-1.deb	43.8 MB
 Ethereum-Wallet-linux32-0-11-1.zip	65.5 MB
 Ethereum-Wallet-linux64-0-11-1.deb	42.2 MB
 Ethereum-Wallet-linux64-0-11-1.zip	63.4 MB
 Ethereum-Wallet-macosx-0-11-1.dmg	67.2 MB

下载 dev 文件，下载完毕直接点击即可安装。

2>私有链创世区块搭建

1.新建文件 piccgenesis.json,输入如下内容并保存；

```
1 {
2     "config": {
3         "chainId": 10,
4         "homesteadBlock": 0,
5         "eip155Block": 0,
6         "eip158Block": 0
7     },
8     "nonce": "0x0000000000000042",
9     "mixhash": "0x000000000000000000000000000000000000000000000000",
10    "difficulty": "0x4000",
11    "alloc": {},
12    "coinbase": "0x00000000000000000000000000000000",
13    "timestamp": "0x00",
14    "parentHash": "0x000000000000000000000000000000000000000000000000",
15    "extraData": "",
16    "gasLimit": "0xffffffff"
17 }
```

各字段含义这里不多做解释，网上教程很详细；

2.搭建私有链

进入 piccgenesis.json 文件所在文件夹，终端输入命令：

```
privatechain$ geth --datadir chaindata init piccgenesis.json
```

```
INFO [11-01|08:54:43.513] Successfully wrote genesis state database=lightchaindata
hash=6e92f8...23a660
```

```
Fatal: invalid genesis file: invalid character 'Â' looking for beginning of object key string
```

```
privatechain$ geth --datadir chaindata --networkid 1002 console
```

```
Welcome to the Geth JavaScript console!
```

[illegible]

- ◆ difficulty：当前区块的挖矿难度；此字段值会随着链的长度的增加而不断增大；
- ◆ extraData：打包区块的人填写的字段，类似于比特币的 coinbase 字段；
- ◆ gasLimit：当前区块允许使用的最大 gas；
- ◆ gasUsed：当前区块累计使用的 gas 总和；
- ◆ hash：区块的哈希串；当这个区块处于 Pending 时此字段返回 null；

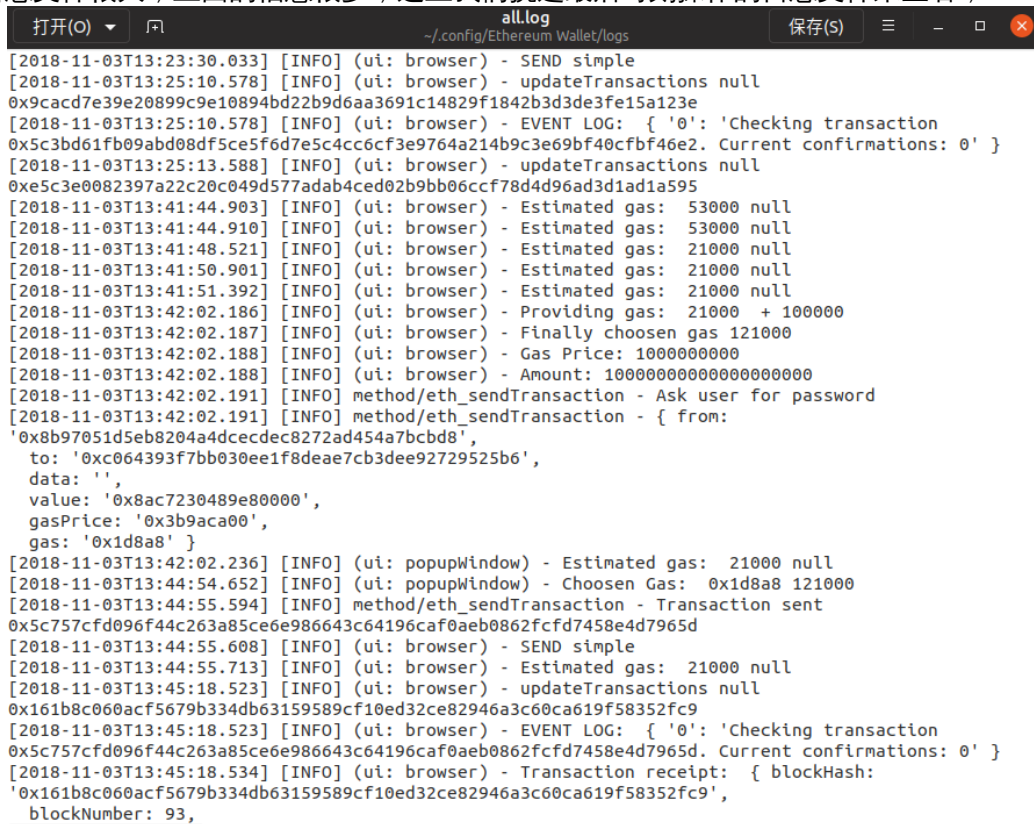
- ◆ logsBloom：当前区块的 Bloom 日志过滤器；当这个区块处于 Pending 时此字段返回 null；
- ◆ miner：打包这个区块的矿工；
- ◆ mixHash：与 nonce 配合用于挖矿，由上一个区块的一部分生成的 hash。注意他和 nonce 的设置需要满足以太坊的黄皮书描述条件；
- ◆ nonce：nonce 就是一个 64 位随机数，用于挖矿，注意他和 mixhash 的设置需要满足以太坊的黄皮书描述的条件；
- ◆ number：区块号；当这个区块处于 pending 此字段返回 null；
- ◆ parentHash：32 字节的父区块的哈希值；
- ◆ receiptsRoot：包含此区块所列的所有交易收据的树的根节点 Hash 值；
- ◆ sha3Uncles：叔区块的哈希值；
- ◆ size：当前这个块的字节大小；
- ◆ stateRoot：区块的最终状态前缀树的根；
- ◆ timestamp：区块打包时的 unix 时间戳；
- ◆ totalDifficulty：区块链到当前块的总难度；
- ◆ transactions：交易对象；或者是 32 字节的交易哈希；
- ◆ transactionsRoot：区块的交易前缀树的根；
- ◆ uncles：叔哈希的数组；

三、对日志输出进行解释

首先找到日志在哪，因为我是用的 Ethereum Wallet 与 geth,所以我直接在 Ethereum Wallet 中打开日志文件，步骤如下：

开发 → 显示日志文件 → all.log

日志文件很大，里面的信息很多，这里我们挑选最后时刻操作的日志文件来查看；



```

[2018-11-03T13:23:30.033] [INFO] (ui: browser) - SEND simple
[2018-11-03T13:25:10.578] [INFO] (ui: browser) - updateTransactions null
0x9cacd7e39e20899c9e10894bd22b9d6aa3691c14829f1842b3d3de3fe15a123e
[2018-11-03T13:25:10.578] [INFO] (ui: browser) - EVENT LOG: { '0': 'Checking transaction
0x5c3bd61fb09abd08df5ce5f6d7e5c4cc6cf3e9764a214b9c3e69bf40cfbf46e2. Current confirmations: 0' }
[2018-11-03T13:25:13.588] [INFO] (ui: browser) - updateTransactions null
0xe5c3e0082397a22c20c049d577adab4ced02b9bb06ccf78d4d96ad3d1ad1a595
[2018-11-03T13:41:44.903] [INFO] (ui: browser) - Estimated gas: 53000 null
[2018-11-03T13:41:44.910] [INFO] (ui: browser) - Estimated gas: 53000 null
[2018-11-03T13:41:48.521] [INFO] (ui: browser) - Estimated gas: 21000 null
[2018-11-03T13:41:50.901] [INFO] (ui: browser) - Estimated gas: 21000 null
[2018-11-03T13:41:51.392] [INFO] (ui: browser) - Estimated gas: 21000 null
[2018-11-03T13:42:02.186] [INFO] (ui: browser) - Providing gas: 21000 + 100000
[2018-11-03T13:42:02.187] [INFO] (ui: browser) - Finally choosen gas 121000
[2018-11-03T13:42:02.188] [INFO] (ui: browser) - Gas Price: 1000000000
[2018-11-03T13:42:02.188] [INFO] (ui: browser) - Amount: 10000000000000000000
[2018-11-03T13:42:02.191] [INFO] method/eth_sendTransaction - Ask user for password
[2018-11-03T13:42:02.191] [INFO] method/eth_sendTransaction - { from:
'0x8b97051d5eb8204a4dcecdcc8272ad454a7bcbdb8',
  to: '0xc064393f7bb030ee1f8deae7cb3dee92729525b6',
  data: '',
  value: '0x8ac7230489e80000',
  gasPrice: '0x3b9aca00',
  gas: '0x1d8a8' }
[2018-11-03T13:42:02.236] [INFO] (ui: popupWindow) - Estimated gas: 21000 null
[2018-11-03T13:44:54.652] [INFO] (ui: popupWindow) - Choosen Gas: 0x1d8a8 121000
[2018-11-03T13:44:55.594] [INFO] method/eth_sendTransaction - Transaction sent
0x5c757cfd096f44c263a85ce6e986643c64196caf0aeb0862fcfd7458e4d7965d
[2018-11-03T13:44:55.608] [INFO] (ui: browser) - SEND simple
[2018-11-03T13:44:55.713] [INFO] (ui: browser) - Estimated gas: 21000 null
[2018-11-03T13:45:18.523] [INFO] (ui: browser) - updateTransactions null
0x161b8c060acf5679b334db63159589cf10ed32ce82946a3c60ca619f58352fc9
[2018-11-03T13:45:18.523] [INFO] (ui: browser) - EVENT LOG: { '0': 'Checking transaction
0x5c757cfd096f44c263a85ce6e986643c64196caf0aeb0862fcfd7458e4d7965d. Current confirmations: 0' }
[2018-11-03T13:45:18.534] [INFO] (ui: browser) - Transaction receipt: { blockHash:
'0x161b8c060acf5679b334db63159589cf10ed32ce82946a3c60ca619f58352fc9',
  blockNumber: 93,

```

由以上截图，日志的第一个括号里面是指令的执行时间，第二个括号是显示指令类型：
 INFO：提示信息；WARN：警告；ERROR：错误提示；
 第三个字段是具体操作，或者注释（根据不同的指令会有不同的显示）；

第四个字段，在这里的截图里，是交易的各字段的名称和值；

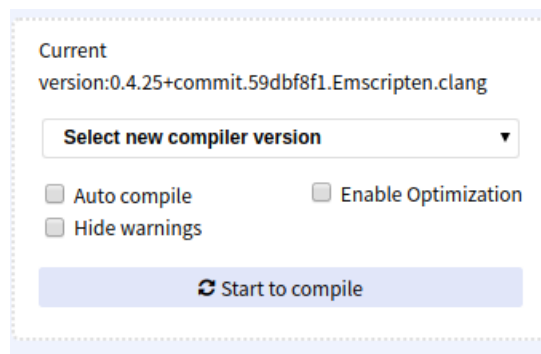
注意，在不同的情况下，日志的第三个字段和第四个字段的类型很有可能是不同的，要根据具体情况具体分析。

四、编写简单的智能合约，在 remix 下进行调试，并部署在链上进行调用

1>编写简单的智能合约

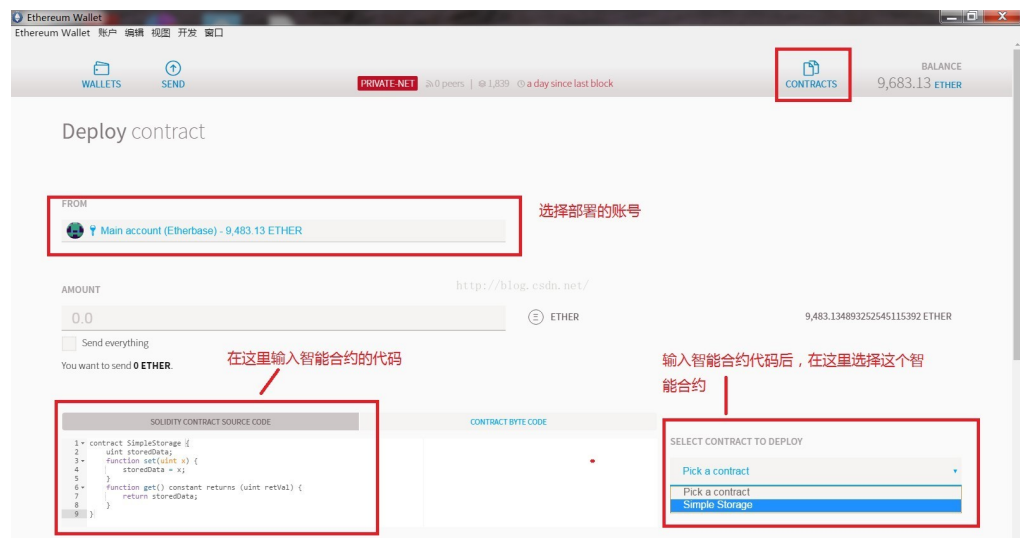
```
browser/MyContract.sol x
1 contract MyContract{
2
3   uint storedData;
4
5   function set(uint x) public
6   {
7
8     storedData = x;
9
10  }
11
12  function get() public constant returns (uint) {
13
14    return storedData;
15
16  }
17
18 }
19
```

2>在 remix 进行调试，点击 Start to compile，没有报错，说明程序没有错误；

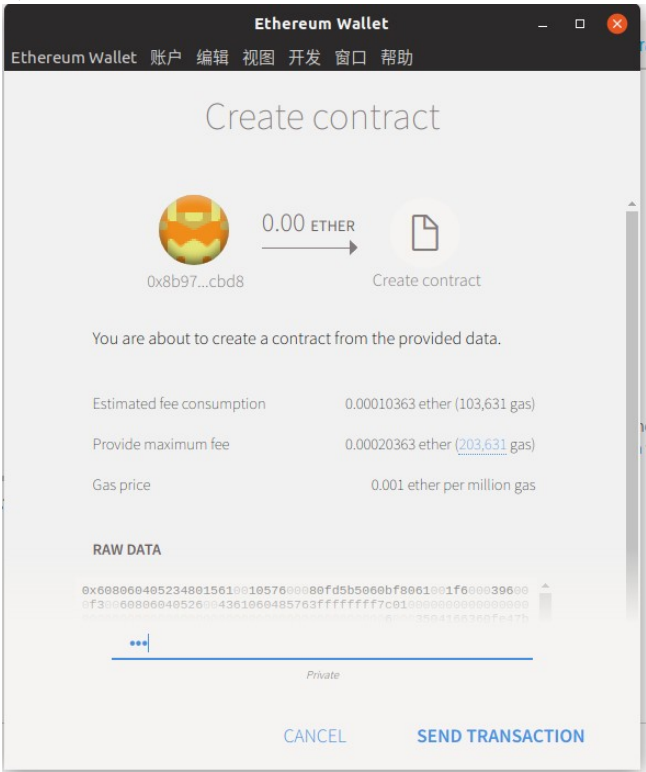


3>部署到链上进行调用

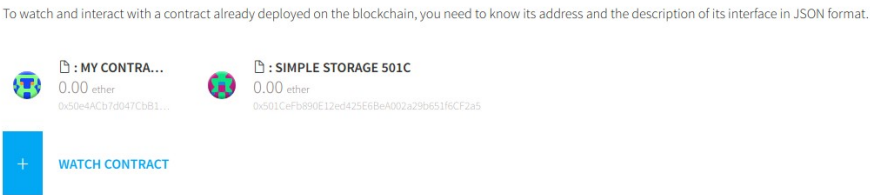
Ethereum Wallet 启动页，点击 CONTRACTS → DEPLOY NEW CONTRACT:
弹出页面进行操作，明细如下：



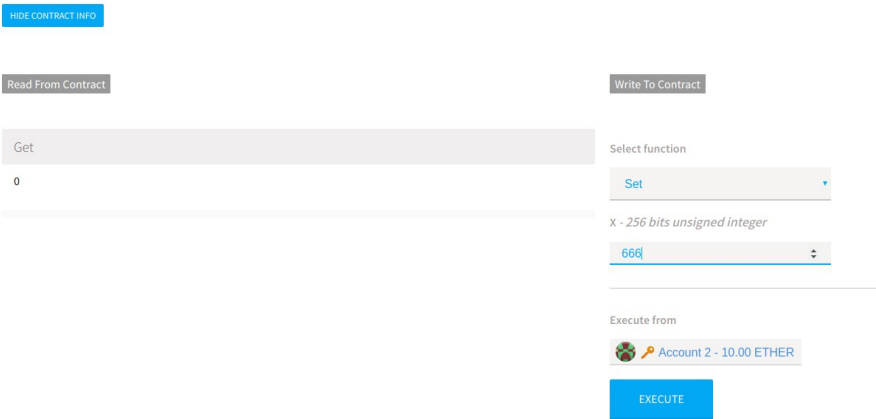
输入密码确认部署：



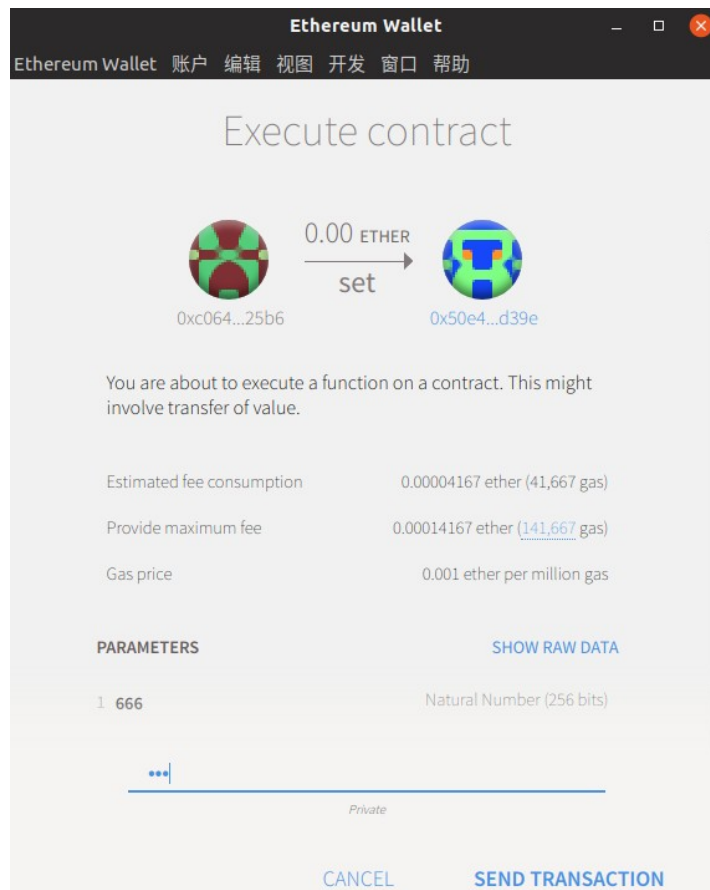
部署成功之后并不能马上看到合约，需要再挖 12 个矿即可显示，如下图所示：



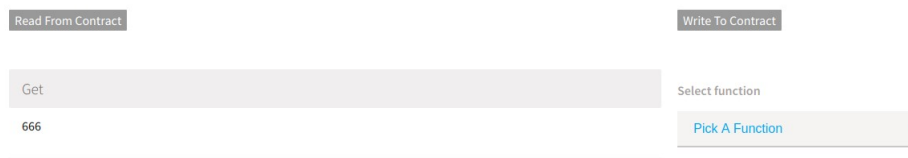
对合约进行调用，这里我们调用合约的 set 函数，将 storedData 的值设置为 666，如下图所示：



点击 EXECUTE 按钮即可执行，按下按钮之后会弹出确认框，输入密码，点击 send transaction,则交易发生；如下图所示：



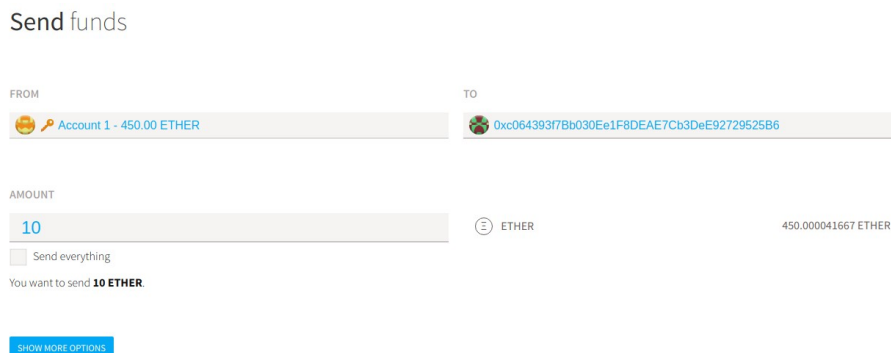
send transaction 之后 get 函数显示的 storedData 的值并不会马上变为 666，依旧是 0，此时在 geth 开启挖矿，挖到矿后可以复查 get 函数显示的 storedData 函数的值变为 666，如下图所示：



到这里我们就实现了一个简单的交易了。

五、对交易的字段进行解释

首先我们创建一笔交易，Account 1 转账 10 个 ETH 给 Account 2，如下图所示：



若等待一会还不行，再行键入 `miner.start(1)`（再不行可以考虑换个环境，我最开始使用 windows 不行后来改成 ubuntu 就可以了）

2.ubuntu 系统下终端启动 geth 客户端，但是打开 Ethereum Wallet 或 Mist,终端下的操作完全不可见；

解决办法：windows 下打开 Ethereum Wallet 或 Mist 默认打开的是私链，但是 ubuntu

下会打开两个不同的 geth,解决的办法是不要通过 GUI 打开 Ethereum Wallet, 打开一个与启动 geth 节点不同的新的终端，键入：`ethereumwallet/mist -rpc` 你的 `geth.ipc` 的文件路径/`geth.ipc`；

3.键入 `ethereumwallet/mist -rpc` 你的 `geth.ipc` 的文件路径/`geth.ipc` 显示连接失败；

解决办法：检查你存放私链数据的文件夹是否放在桌面，若是，移到别的位置；

检查你的 `geth.ipc` 的文件路径上是否有中文，若有，改掉即可；