以太坊私网建立 、合约编译、部署完全教程（1）

> 孔壹学院：国内区块链职业教育领先品牌
>
> 作者：黎跃春，区块链、高可用架构工程师
> 微信：liyc1215 QQ群：348924182 博客：http://liyuechun.org

一、为什么用到私有链?

在以太坊的共有链上部署智能合约、发起交易需要花费以太币。而通过修改配置，可以在本机搭建一套以太坊私有链，因为与公有链没关系，既不用同步公有链庞大的数据，也不用花钱购买以太币，很好地满足了智能合约开发和测试的要求，开发好的智能合约也可以很容易地切换接口部署到以太坊公有链上。

二、开源工具和语言

1、brewMacOS包管理器

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

2、install Go compiler

```
liyuechun:Downloads yuechunli$ brew install go
```

3、geth运行以太坊节点

下载Source code (tar.gz)

```
liyuechun:Downloads yuechunli$ cd go-ethereum-1.5.9
liyuechun:go-ethereum-1.5.9 yuechunli$ pwd
/Users/liyuechun/Downloads/go-ethereum-1.5.9
liyuechun:go-ethereum-1.5.9 yuechunli$ make geth
```

4、Solidity以太坊智能合约语言

```
brew update
brew upgrade
brew tap ethereum/ethereum
brew install solidity
brew linkapps solidity
```

备注：安装时间可能有点长，请耐心等待...
备注：安装时间可能有点长，请耐心等待...
备注：安装时间可能有点长，请耐心等待...

如果碰见下面的错误，请移步：http://blog.csdn.net/Sico2Sico/article/details/71082130

```
The GitHub credentials in the macOS keychain may be invalid.
Clear them with:
  printf "protocol=https\nhost=github.com\n" | git credential-osxkeychain er
ase
Or create a personal access token:
  https://github.com/settings/tokens/new?scopes=gist,public_repo&description
=Homebrew
```

三、建立私链

1. 创建一个文件夹来存储你的私链数据

```
liyuechun:1015 yuechunli$ mkdir privatechain
liyuechun:1015 yuechunli$ pwd
/Users/liyuechun/Desktop/1015
liyuechun:1015 yuechunli$ ls
privatechain
liyuechun:1015 yuechunli$
```

2. 使用 `geth` 来加载

```
geth --networkid 123 --dev --datadir data1 --rpc --rpcaddr 192.168.1.5 --rpc
port 8989 --port 3000
```

各选项含义如下：

- `--identity:` 指定节点 ID；
- `--rpc:` 表示开启 HTTP-RPC 服务；
- `--rpcaddr:` HTTP-RPC 服务ip地址；
- `--rpcport:` 指定 HTTP-RPC 服务监听端口号（默认为 8545）；
- `--datadir:` 指定区块链数据的存储位置；
- `--port:` 指定和其他节点连接所用的端口号（默认为 30303）；
- `--nodiscover:` 关闭节点发现机制，防止加入有同样初始配置的陌生节点。

执行上面的命令，你应该能看到下面的信息：

> INFO [10-15|03:14:50] IPC endpoint opened:
> /Users/liyuechun/Desktop/1015/privchain/geth.ipc
> INFO [10-15|03:14:50] HTTP endpoint opened: http://127.0.0.1:8545

如果你切换到 `privchain` 文件夹里面，你会看到 `geth`，`geth.ipc`，和 `keystore`。

```
liyuechun:1015 yuechunli$ cd data1/
liyuechun:data1 yuechunli$ ls
geth        geth.ipc    keystore
liyuechun:data1 yuechunli$
```

- 保持节点的运行，不要关闭终端，重新打开一个终端，使用 `geth attach` 连接节点，并且打开 `geth console`

```
liyuechun:privchain yuechunli$ geth attach ipc:/Users/liyuechun/Desktop/1015
/privchain/geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.7.1-stable-05101641/darwin-amd64/go1.9.1
 modules: admin:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0
 shh:1.0 txpool:1.0 web3:1.0

>
```

这是一个交互式的 JavaScript 执行环境，在这里面可以执行 JavaScript 代码，其中 > 是命令提示符。在这个环境里也内置了一些用来操作以太坊的 JavaScript 对象，可以直接使用这些对象。这些对象主要包括：

- `eth:` 包含一些跟操作区块链相关的方法；
- `net:` 包含一些查看p2p网络状态的方法；

- `admin:` 包含一些与管理节点相关的方法；
- `miner:` 包含启动&停止挖矿的一些方法；
- `personal:` 主要包含一些管理账户的方法；
- `txpool:` 包含一些查看交易内存池的方法；
- `web3:` 包含了以上对象，还包含一些单位换算的方法。

3. 相关api命令

查看账户

```
> personal.listAccounts
[]
>
```

创建账户

```
> personal.newAccount('liyuechun')
"0xb6d7d842e7dc9016fa6900a183b2be26fc90b2d8"
>
```

PS：里面的`liyuechun`是你账户的密码，输入你自己喜欢的密码。

查看账户

```
> personal.listAccounts
["0xb6d7d842e7dc9016fa6900a183b2be26fc90b2d8"]
>
```

4. web3命令

https://ethereumbuilders.gitbooks.io/guide/content/en/ethereum_javascript_api.html

```
> web3.eth.coinbase
"0xb6d7d842e7dc9016fa6900a183b2be26fc90b2d8"
>
```

5. 编写智能合约代码

```
pragma solidity ^0.4.4;
```

```
contract test {

    function multiply(uint a) returns(uint d){

        return a * 7;
    }

}
```

6. 获取智能合约字节码和abi

代码拷贝到https://remix.ethereum.org，编译，然后拷贝字节码和ABI。

- 字节码

```
6060604052341561000f57600080fd5b5b60ab8061001e6000396000f3006060604052600035
7c0100000000000000000000000000000000000000000000000000000000900463ffffffff16
8063c6888fa114603d575b600080fd5b341560475760008080359060200190
91905050607156 5b6040518082815260200191505060405180910390f35b6000600782029050
5b9190505600a165627a7a7230582067d7c851e14e862886b6f53dad6825135557fb3a4b6913
50c94ea5b80605f6770029
```

- ABI

```
[
  {
    "constant": true,
    "inputs": [
      {
        "name": "a",
        "type": "uint256"
      }
    ],
    "name": "multiply",
    "outputs": [
      {
        "name": "d",
        "type": "uint256"
      }
    ],
    "payable": false,
    "type": "function",
    "stateMutability": "view"
  }
```

```
]
```

7. 在bejson中转义成字符串

http://www.bejson.com

```
[{\"constant\":true,\"inputs\":[{\"name\":\"a\",\"type\":\"uint256\"}],\"nam
e\":\"multiply\",\"outputs\":[{\"name\":\"d\",\"type\":\"uint256\"}],\"payab
le\":false,\"type\":\"function\",\"stateMutability\":\"view\"}]
```

7. 通过abi创建合约对象

```
> var abi = JSON.parse('[{\"constant\":true,\"inputs\":[{\"name\":\"a\",\"ty
pe\":\"uint256\"}],\"name\":\"multiply\",\"outputs\":[{\"name\":\"d\",\"type
\":\"uint256\"}],\"payable\":false,\"type\":\"function\",\"stateMutability\"
:\"view\"}]')
> myContract = web3.eth.contract(abi)
{
  abi: [{
      constant: false,
      inputs: [{...}],
      name: "multiply",
      outputs: [{...}],
      payable: false,
      type: "function"
  }],
  eth: {
    accounts: ["0x2abf46d8b0d940cdeedd55872bc0648add40227d"],
    blockNumber: 384,
    coinbase: "0x2abf46d8b0d940cdeedd55872bc0648add40227d",
    compile: {
      lll: function(),
      serpent: function(),
      solidity: function()
    },
    defaultAccount: undefined,
    defaultBlock: "latest",
    gasPrice: 0,
    hashrate: 0,
    mining: false,
    pendingTransactions: [],
    protocolVersion: "0x3f",
    syncing: false,
    call: function(),
    contract: function(abi),
```

```
    estimateGas: function(),
    filter: function(fil, callback),
    getAccounts: function(callback),
    getBalance: function(),
    getBlock: function(),
    getBlockNumber: function(callback),
    getBlockTransactionCount: function(),
    getBlockUncleCount: function(),
    getCode: function(),
    getCoinbase: function(callback),
    getCompilers: function(),
    getGasPrice: function(callback),
    getHashrate: function(callback),
    getMining: function(callback),
    getPendingTransactions: function(callback),
    getProtocolVersion: function(callback),
    getRawTransaction: function(),
    getRawTransactionFromBlock: function(),
    getStorageAt: function(),
    getSyncing: function(callback),
    getTransaction: function(),
    getTransactionCount: function(),
    getTransactionFromBlock: function(),
    getTransactionReceipt: function(),
    getUncle: function(),
    getWork: function(),
    iban: function(iban),
    icapNamereg: function(),
    isSyncing: function(callback),
    namereg: function(),
    resend: function(),
    sendIBANTransaction: function(),
    sendRawTransaction: function(),
    sendTransaction: function(),
    sign: function(),
    signTransaction: function(),
    submitTransaction: function(),
    submitWork: function()
  },
  at: function(address, callback),
  getData: function(),
  new: function()
}
```

8. 检查coinbase账号余额

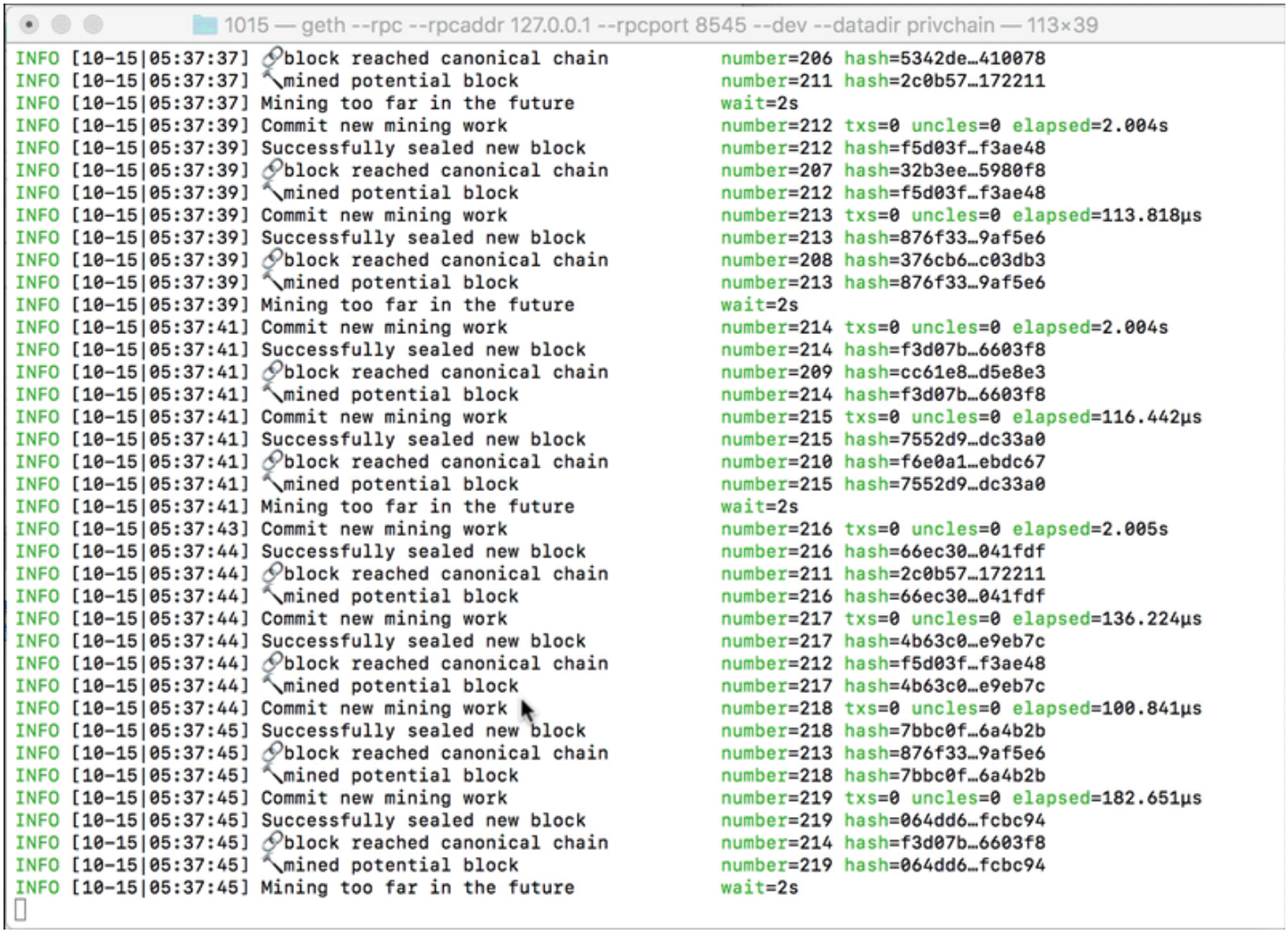
```
> account1 = web3.eth.coinbase
```

```
"0x2abf46d8b0d940cdeedd55872bc0648add40227d"
> web3.eth.getBalance(account1)
0
>
```

如果余额大于0，继续，否则，开始挖矿。

```
> miner.start();
null
>
```

挖矿过程中，切换到节点终端，你会发现一直在挖矿。



如果你觉得差不多了，可以运行下面的命令停止挖矿。

```
miner.stop();
```

9. 停止挖矿，并且查余额

```
> miner.start();
null
> miner.stop();
true
> web3.eth.getBalance(account1)
1.152e+21
>
```

10. 解锁coinbase账号，我们通过coinbase账号来付费部署合约

`liyuechun`：换成你的密码。

```
> personal.unlockAccount(account1, 'liyuechun')
true
>
```

11. 预估手续费

```
> bytecode = "6060604052341561000f57600080fd5b5b60ab8061001e6000396000f30060
606040526000357c0100000000000000000000000000000000000000000000000000000090
0463ffffffff168063c6888fa114603d575b600080fd5b3415604757600080fd5b605b600480
8035906020019091905050606071565b60405180828152602001915050604051809103 90f35b60
006007820290505b9190505600a165627a7a7230582067d7c851e14e862886b6f53dad682513
5557fb3a4b691350c94ea5b80605f6770029"
"6060604052341561000f57600080fd5b5b60ab8061001e6000396000f300606060405260003
57c0100000000000000000000000000000000000000000000000000000000900463ffffffff1
68063c6888fa114603d575b600080fd5b3415604757600080fd5b605b6004808035906020019
091905050606071565b60405180828152602001915050604051809103 90f35b600060078202905
05b9190505600a165627a7a7230582067d7c851e14e862886b6f53dad6825135557fb3a4b691
350c94ea5b80605f6770029"
> web3.eth.estimateGas({data: bytecode})
Error: invalid argument 0: json: cannot unmarshal hex string without 0x pref
ix into Go struct field CallArgs.data of type hexutil.Bytes
    at web3.js:3104:20
    at web3.js:6191:15
    at web3.js:5004:36
    at <anonymous>:1:1

> bytecode = "0x6060604052341561000f57600080fd5b5b60ab8061001e6000396000f300
6060606040526000357c0100000000000000000000000000000000000000000000000000000
900463ffffffff168063c6888fa114603d575b600080fd5b3415604757600080fd5b605b6004
8080359060200190919050506071565b60405180828152602001915050604051809103 90f35b
60006007820290505b9190505600a165627a7a7230582067d7c851e14e862886b6f53dad6825
135557fb3a4b691350c94ea5b80605f6770029"
```

```
"0x6060604052341561000f57600080fd5b5b60ab8061001e6000396000f3006060604052600
0357c0100000000000000000000000000000000000000000000000000000000900463ffffffff
f168063c6888fa114603d575b600080fd5b341560475760080fd5b605b60048080359060200
190919050506071565b60405180828152602001915050604051809103905f35b6000600782029
0505b9190505600a165627a7a7230582067d7c851e14e862886b6f53dad6825135557fb3a4b6
91350c94ea5b80605f6770029"
> web3.eth.estimateGas({data: bytecode})
98391
>
```

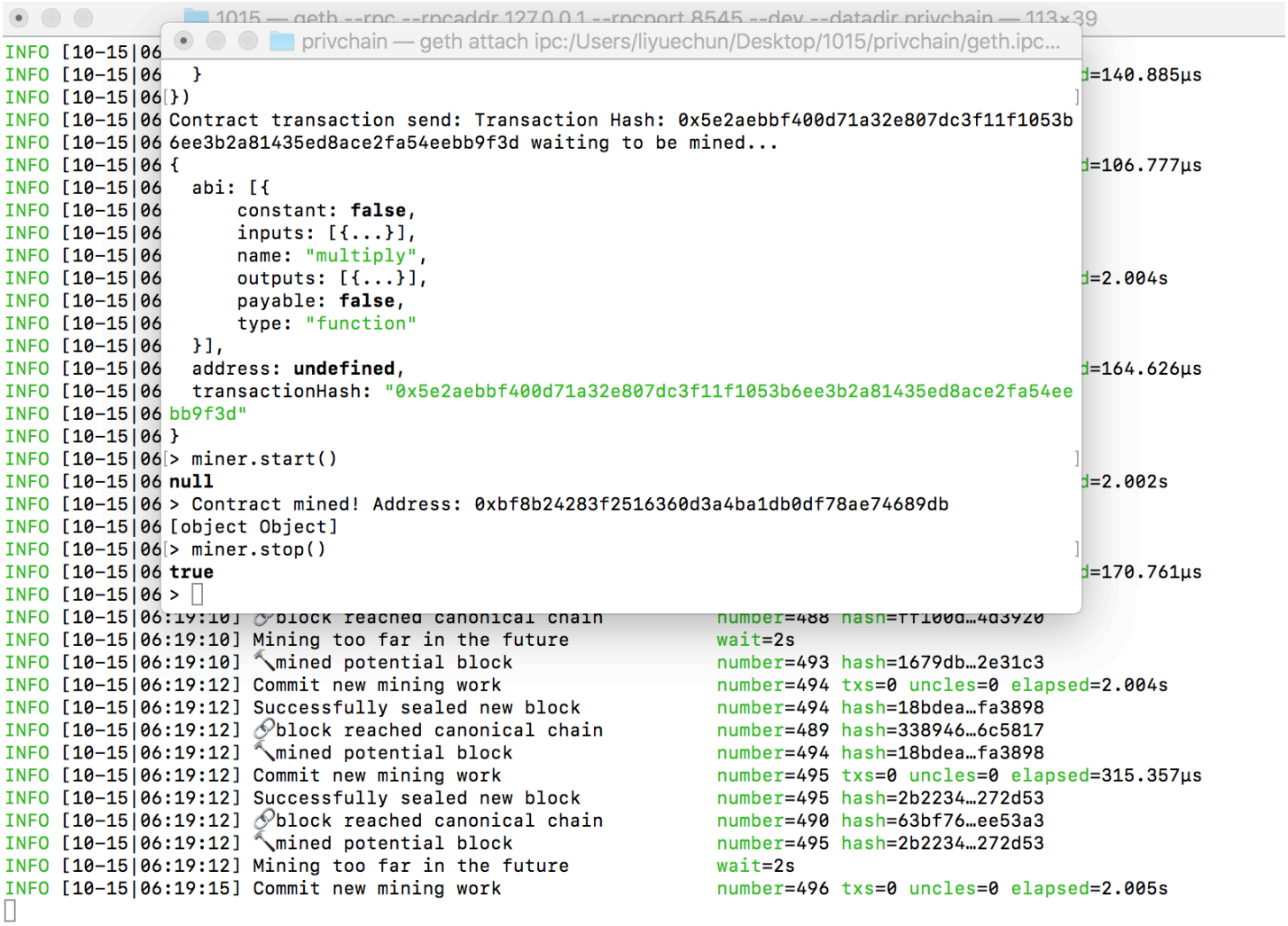
备注：字节码前面需要添加 `0x` 。手续费大概为 `98391 gas` 。

12. 部署合约，为了方便理解，设置一个回调函数

```
> contractInstance = contract.new({data: bytecode, gas: 1000000, from: accou
nt1}, function(e, contract){
  if(!e){
    if(!contract.address){
      console.log("Contract transaction send: Transaction Hash: "+contract.t
ransactionHash+" waiting to be mined...");
    }else{
      console.log("Contract mined! Address: "+contract.address);
      console.log(contract);
    }
  }else{
    console.log(e)
  }
})
Contract transaction send: Transaction Hash: 0x5e2aebbf400d71a32e807dc3f11f1
053b6ee3b2a81435ed8ace2fa54eebb9f3d waiting to be mined...
{
  abi: [{
      constant: false,
      inputs: [{...}],
      name: "multiply",
      outputs: [{...}],
      payable: false,
      type: "function"
  }],
  address: undefined,
  transactionHash: "0x5e2aebbf400d71a32e807dc3f11f1053b6ee3b2a81435ed8ace2fa
54eebb9f3d"
}
>
```

13. 你的合约等待挖矿，开始挖矿，等一会儿，停止

```
> miner.start()
null
> Contract mined! Address: 0xbf8b24283f2516360d3a4ba1db0df78ae74689db
[object Object]
> miner.stop()
true
>
```



14. 检查合约是否部署成功

```
> eth.getCode(contractInstance.address)
"0x60606040526000357c0100000000000000000000000000000000000000000000000000000000
000900463ffffffff168063c6888fa114603d575b600080fd5b3415604757600080fd5b605b6
0048080359060200190919050506071565b60405180828152602001915050604051809103 90f
35b6000600782029050 5b9190505600a165627a7a7230582067d7c851e14e862886b6f53dad6
825135557fb3a4b691350c94ea5b80605f6770029"
>
```

15. 调用合约方法

```
> contractInstance.multiply(6)
42
>
```

PS： 这里添加`call`的原因是因为`multiply`函数没有添加`constant`。

```
pragma solidity ^0.4.4;

contract test {

    function multiply(uint a) returns(uint d){

        return a * 7;
    }

}
```

Over Game!!!!

技术交流

- 区块链技术交流QQ群：348924182
- 「区块链部落」官方公众号

长按，识别二维码，加关注