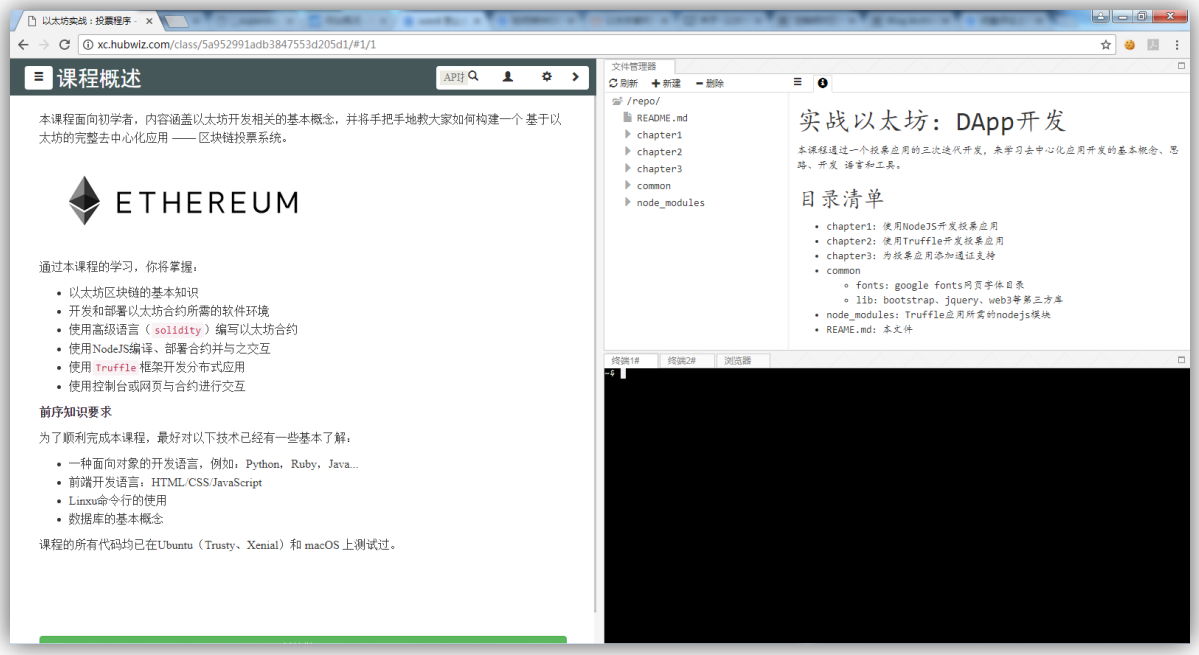
导读

本电子书由汇智网（http://www.hubwiz.com）编目整理，原文由网友趁风卷创作，最早发布于简书社区。

汇智网推出了在线交互式以太坊 DApp 实战开发课程，以去中心化投票应用（Voting DApp）为课程项目，通过三次迭代开发过程的详细讲解与在线实践，并且将区块链的理念与去中心化思想贯穿于课程实践过程中，为希望快速入门区块链开发的开发者提供了一个高效的学习与价值提升途径。读者可以通过以下链接访问《以太坊 DApp 开发实战入门》在线教程：

http://xc.hubwiz.com/course/5a952991adb3847553d205d1?affid=geth7878

教程预置了开发环境。进入教程后，可以在每一个知识点立刻进行同步实践，而不必在开发环境的搭建上浪费时间：



geth 使用指南

本文是对以太坊文档 Ethereum Frontier Guide 和 Ethereum Homestead 的整理。Fontier 和 Homestead 分别是以太坊的第 1 和 第 2 个版本。

本文使用 go 语言编写的命令行客户端 geth

geth 的命令可以分为 CLI 命令和 JavaScript 控制台交互式命令，约定如下

- geth account list：这是 CLI 命令
- > eth.accounts：这是 javaScript 控制台的命令，前面有一个 >。进入控制台的方式为 geth console

1. 安装和运行节点

1.1 安装客户端

geth 的安装教程请参见 Building Ethereum

1.2 同步区块

「同步」的意思是把网络上的区块全部下载到本地，以同步到网络的最新状态。使用客户端前必须先同步区块。

同步命令如下

- geth：全节点模式同步模式
- geth --fast --cache=1024：--fast 快速同步模式，只下载状态（state downloads）

更优雅的同步方式

- geth --fast console 2>network_sync.log：同步时把输出日志重定向到 network_sync.log 中，并进入控制台。这样就可以边同步边使用控制台命令。
- 用 tail -f network_sync.log 可以重新浏览到日志

数据存放目录

主网络区块数据的默认存放目录是 ~/Library/Ethereum（Mac OS X）

- 其他系统下，可用该方式找到默认路径：geth -h 后搜索 --datadir，后面跟着的就是默认目录
- 如果你想将区块下载到其他目录，可以使用命令 geth --fast --datadir "<path>"

不同客户端是可以共用区块数据的。用 geth 同步的区块数据，可以在 Ethereum Wallet 或 Mist 客户端里直接使用。

导入已有的区块文件

如果本地已有区块文件，可以将其导入

- geth export filename：导出区块文件
- geth import filename：导入区块文件

启动节点

- geth 借助启动节点（bootstrap nodes）来初始化寻找过程。启动节点被写在源码里，但可用这些方式修改
  - geth --bootnodes "enode://pubkey1@ip1:port1 enode://pubkey2@ip2:port2 enode://pubkey3@ip3:port3"，pubkey、ip 和 port 依次为启动节点的公钥地址、ip 和端口号。
  - > admin.addPeer("enode://pubkey@ip:port")
- geth 使用名为 discover protocol 的协议来寻找其他节点

1.3 启动客户端

启动客户端的方式如下

- 主网络
  - 如果区块数据在默认目录下：geth
  - 如果区块数据在其他目录下：geth --datadir <path>
- 测试网络：geth --datadir <path> --networkid 15。你只会连接与你的协议版本和 networkid 都相同的节点。主网络的 networkid 是 1，所以 networkid 只要不是 1 就可以

更常用的是启动客户端，并进入控制台模式：geth --datadir <path> console 2>console.log。同时可以另开窗口，用 tail -f console.log 浏览日志。

更复杂的启动命令

geth --identity "MyNodeName" --rpc --rpcport "8080" --rpccorsdomain "*" --datadir "./ethdev" --port "30303" --nodiscover --rpcapi "db,eth,net,web3" --networkid 1999（参考：Command line parameters）

- identity "MyNodeName"：为你的节点设置身份标识，以更容易在节点列表中便是
- --rpc：开启 RPC 接口
- --rpcport "8080"：RPC 端口
- --rpccorsdomain "*"：设置能连接到你的节点的 URL，用来完成 RPC 任务。* 指任何 URL 都能连接到你。
- --datadir "./ethdev"：区块数据文件夹
- --port "30303"：用来监听其他节点的端口
- --nodiscover：你的节点不会被其他人发现，除非他们手动添加你
- --rpcapi "db,eth,net,web3"：提供给别人使用的 RPC API，默认为 web3 接口
- networkid 1999：相同 networkdid 才会连接到一起

监控

在控制台里，使用这些命令检查连接状态

- > net.listening：检查是否连接
- > net.peerCount：连接到的节点个数
- > admin.peers：返回连接到的节点的详细信息
- > admin.nodeInfo：返回本地节点的详细信息

此外，还有一些网站供你查看以太坊主网络的状态

- EthStats.net：查看以太坊网络的实时状态。本地安装教程 Eth-Netstats README on Github
- EtherNodes.com：查看节点数据
- etherchain.org：另一个查看实时以太坊状态的网站

1.4 测试网络

以太坊公有的测试网络有 Ropsten 和 Rinkeby。除此之外，你可以搭建自己的私有网络，即只能本地访问的私网。

下面介绍 3 种测试网络的搭建方式

Ropsten 网络

- 同步区块：geth --testnet --fast --bootnodes "enode://20c9ad97c081d63397d7b685a412227a40e23c8bdc6688c6f37e97cfbc22d2b4d1db1510d8f61e6a8866ad7f0e17c02b14182d37ea7c3c8b9c2683aeb6b733a1@52.169.14.227:30303,enode://6ce05930c72abc632c58e2e4324f7c7ea478cec0ed4fa2528982cf34483094e9cbc9216e7aa349691242576d552a2a56aaeae426c5303ded677ce455ba1acd9d@13.84.180.240:30303",

来连接特殊的启动节点来同步 Ropsten 网络的数区块。或者也可以使用 Mist 进行同步。

- 进入网络：geth --testnetwork

参考：ethereum/ropsten

Rinkeby 网络

参见 Rinkeby: Ethereum Testnet - Connect Yourself，有 archive, full, light, embedded 4 种模式

私有网络

搭建私有网络，需要先新建创世块参数文件 genesis.json

```
{
    "config": {
        "chainId": 15,
        "homesteadBlock": 0,
        "eip155Block": 0,
        "eip158Block": 0
    },
    "difficulty": "10000",
    "gasLimit": "2100000",
    "alloc": {
        "7df9a875a174b3bc565e6424a0050ebc1b2d1d82": { "balance": "300000" },
        "f41c74c9ae680c1aa78f42e5647a62f353b7bdde": { "balance": "400000" }
    }
}
```

接下来使用命令 geth --datadir ./ethdev init <genesis.json> console 初始化测试网络，并进入控制台。

该测试网络只有你一个人，你需要自己挖坑来记录交易。

参考

- Test Networks
- Private network

2. 账户管理

2.1 创建账户

新建账户

- geth account new
- > personal.newAccount("passphrase")
- 非交互方式：geth --password <passwordfile> account new，密码以明文方式写在文件 passwordfile 里

导入账户

- geth account import <keyfile>，私钥以明文方式写在文件 keyfile 里，每个一行。
  - <keyfile> 是存着账户信息的 json 文件，其中私钥是被账号密码（password）加密后，存放在里面的
  - 不同平台的 keyfile 默认存储位置是不同的
    - Windows: C:\Users\username%appdata%\Roaming\Ethereum\keystore
    - Linux: ~/.ethereum/keystore
    - Mac: ~/Library/Ethereum/keystore
- 结合 --password，可以使用在导入账户时设置密码：geth --password <passwordfile> account import <keyfile>

修改密码

- geth account update <address>
- geth account update 2：2 是账户的编号，在 geth account list 中可以看到

2.2 导入钱包

- geth wallet import <etherwallet.json>

创建多签名钱包，请参见 Creating a Multi-Signature Wallet in Mist

2.3 查看账户信息

列出所有账户

- geth account list；对应的控制台命令为 > eth.accounts

查看账户余额

- > eth.getBalance(<address>)：列出账户余额，单位是 Wei
- >web3.fromWei(eth.getBalance(<address>), "ether")：列出账户余额，单位是 eth

下面的代码可以打印所有的账户余额

```
function checkAllBalances() {
var i =0;
eth.accounts.forEach( function(e){
     console.log("  eth.accounts["+i+"]: " +  e + " \tbalance: " +
web3.fromWei(eth.getBalance(e), "ether") + " ether");
i++;
})
};
```

小技巧：可以把代码存到文件中。进入 geth 的控制台后，用 > loadScript(<loadfile.js>) 导入文件中的函数。

2.4 发送交易

以发起一个 0.01 个 ether 的转账交易为例

```
> var sender = eth.accounts[0];
> var receiver = eth.accounts[1];
> var amount = web3.toWei(0.01, "ether")

> eth.sendTransaction({from:sender, to:receiver, value: amount, gas:
gasAmount})  //`gas` 不是必须的
```

之后会让你输入密码

或者也可以先用 personal.unlockAccount(sender, <passphrase>) 解锁账户，再发送交易。

3. 挖矿

3.1 介绍

挖矿会有挖矿奖励

以太坊使用 Ethash 的 PoW 算法

3.2 CPU 挖矿

挖矿

- geth --mine --minerthreads=4：--minerthreads 设置并行挖矿的线程数量，默认为所有的处理器数量；
- > miner.star(8)，使用 8 个 minerthreads；>miner.stop() 停止
- 稍微复杂点的挖矿命令：geth --mine --minerthreads 4 --datadir /usr/local/share/ethereum/30303 --port 30303：使用不同数据目录（ethereum/30303）和不同的端口（30303）挖矿

发放奖励：eth.etherbase（也叫 coinbase）是一个地址，挖矿奖励会发到这个地址里。改变 etherbase 的方式如下

- geth --etherbase 1 --mine：改变 etherbase 为编号 1 的地址；或 geth --etherbase '0xa4d8e9cae4d04b093aac82e6cd355b6b963fb7ff'
- 控制台：> miner.setEtherbase(eth.accounts[2])

其他

- > eth.getBlock(i).miner：查看块的挖出者
- eth.getBlockTransactionCount("pending")：查看未确认交易的数量
- eth.getBlock("pending", true).transactions：查看所有未确认交易

下面的函数可以统计不同地址的出块数量

```
function minedBlocks(lastn, addr) {
  addrs = [];
  if (!addr) {
    addr = eth.coinbase
  }
  limit = eth.blockNumber - lastn
  for (i = eth.blockNumber; i >= limit; i--) {
    if (eth.getBlock(i).miner == addr) {
      addrs.push(i)
    }
  }
  return addrs
}
// scans the last 1000 blocks and returns the blocknumbers of blocks mined by your coinbase
// (more precisely blocks the mining reward for which is sent to your coinbase).
minedBlocks(1000, eth.coinbase);
```

//[352708, 352655, 352559]

3.3 其他挖矿方式

- GPU 挖矿
- 矿池挖矿

4. 接口

4.1 命令行接口

CLI 命令已介绍得差不多了。

可以去 Command Line Options 查阅具体的命令。

4.2 JSON RPC API

JSON-RPC 是一种无状态、轻量级的 RPC 协议，规定了通信的数据结构和规则。以太坊客户端使用 JSON-RPC 和其他客户端通信。

比如 MetaMask 钱包就是通过 JSON-RPC 和以太坊客户端进行通信的。

对于不同的以太坊客户端，默认 JSON-RPC 地址如下

- C++: http://localhost:8545
- Go: http://localhost:8545
- Py: http://localhost:4000

geth 是 go 客户端，因此 JSON-RPC 为 http://localhost:8545。

常用命令

- geth --rpc：开启 HTTP JSON-RPC
- geth --rpc --rpcaddr <ip> --rpcport <portnumber>：改变 JSON-RPC 的 ip 和端口
- 控制台下：> admin.startRPC(addr, port)

细节请参阅 JSON RPC

4.3 使用 Ðapp 的 JavaScript API

你可以使用 web3.js 库所提供的对象，来搭建 Ðapp。

细节请参阅 JavaScript API，中文版请参阅 web3.js API 官方文档中文版。

4.4 JavaScript 控制台

一般操作都在控制台模式下进行。

- 进入控制台的方式：geth console
- 启动测试网络，并进入控制台：geth --datadir ./ethdev console

详细命令请参阅 JavaScript Console