一、Geth介绍

Geth是go-ethereum项目的客户端，也是目前使用最广泛的客户端，通过此客户端可以进行基本所有的以太坊区块链相关操作。它是进入以太坊网络（主网络，测试网络或专用网络）的入口点，能够作为完整节点（默认）归档节点（保留所有历史状态）或轻型节点（实时检索数据）运行。它可以被其他进程用作通过在HTTP、WebSocket和/或IPC传输的顶部暴露的JSON RPC端点进入以太网网络。Ethereum Javascript API

在以太坊的公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，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。那么在私有链上开好了智能合约，如何部署到公有链呢？这一步非常简单，只需要有一定的能够支付部署的gas即可，另外只需改一下配置文件web连接的地址，将连接的本机私有链的地址`http://localhost:8545`改为主网或者测试网络的地址即可。

1. 主要功能

1. JavaScript Console：通过后台进行命令操作；
2. Management API：管理相关的API；
3. JSON-RPC server：JSON-RPC相关调用API

无论通过API或console都可以进行以太坊区块链相关操作，比如：账号管理、交易、挖矿、部署智能合约等功能。

2. Geth与Mist区别

Geth即go-ethereum项目，客户端可以通过对接API，目前交易平台常常使用的方式，或直接通过命令行进行操作。

Mist即Ethereum Wallet客户端，主要是为用户提供可视化操作的客户端，下载安装之后通过相应的图形化界面即可进行创建账户、转账、查询余额等操作。

二、geth环境搭建

1. 下载go-ethereum

```
$ mkdir -p $GOPATH/src/github.com/ethereum
$ cd $GOPATH/src/github.com/ethereum
$ git clone https://github.com/ethereum/go-ethereum.git
```

2. 编译

```
$ cd go-ethereum
$ make geth
```

3. 配置

将 `geth` 添加到环境变量中 `vi ~/.bashrc`， 有的配置文件使用的是~/.bashrc，~/.profile，~/.bash_profile

```
export GETH="$GOPATH/src/github.com/ethereum/go-ethereum/build"
export PATH="$PATH:$GETH/bin"
```

然后执行 `source ~/.bashrc`，使配置生效。

检查是否安装成功

```
geth --help
```

如果输出一些帮助提示命令，则说明安装成功。

三、初始化创世区块

1. 配置创世区块

要运行以太坊私有链，需要定义自己的创世区块，创世区块信息写在一个 JSON 格式的配置文件中。首先将下面的内容保存到一个 JSON 文件中，例如 `genesis.json`

```
$ mkdir ~/privatechain
$ cd privatechain
$ mkdir data0
$ vi genesis.json
```

genesis.json 的代码

```
{
  "config": {
        "chainId": 110,
        "homesteadBlock": 0,
        "eip155Block": 0,
        "eip158Block": 0
    },
  "alloc"      : {},
  "coinbase"   : "0x0000000000000000000000000000000000000000",
  "difficulty" : "0x20000",
  "extraData"  : "",
  "gasLimit"   : "0x2fefd8",
  "nonce"      : "0x0000000000000042",
  "mixhash"    :
"0x0000000000000000000000000000000000000000000000000000000000000000",
```

```
  "parentHash" :
"0x0000000000000000000000000000000000000000000000000000000000000000",
  "timestamp"  : "0x00"
}
```

其中，chainID 指定了独立的区块链网络 ID。网络 ID 在连接到其他节点的时候会用到，以太坊公网的网络 ID 是 1，为了不与公有链网络冲突，运行私有链节点的时候要指定自己的网络 ID。不同 ID 网络的节点无法相互连接。配置文件还对当前挖矿难度 difficulty、区块 Gas 消耗限制 gasLimit 等参数进行了设置。

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。**
nonce	nonce就是一个64位随机数，用于挖矿。
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，所以随意填写。

2. 写入创世区块

准备好创世区块配置文件后，需要初始化区块链，将上面的创世区块信息写入到区块链中。首先要新建一个目录用来存放区块链数据，假设新建的数据目录为 ~/privatechain/data0，genesis.json 保存在 ~/privatechain 中，此时目录结构应该是这样的：

```
privatechain
├── data0
└── genesis.json
```

启动Geth即可以启动以太坊的区块链，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：

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

执行初始化命令：

```
geth --datadir data0 init genesis.json
```

上面的命令的主体是 `geth init`，表示初始化区块链，命令可以带有选项和参数，其中 `--datadir` 选项后面跟一个目录名，这里为 `data0`，表示指定数据存放目录为 `data0`， `genesis.json` 是 `init` 命令的参数。

运行上面的命令，会读取 `genesis.json` 文件，根据其中的内容，将创世区块写入到区块链中。如果看到以下的输出内容，说明初始化成功了。

```
INFO [01-29|21:21:13] Maximum peer count                       ETH=25 LES=0
total=25
INFO [01-29|21:21:13] Allocated cache and file handles
database=/home/lixu/privatechain/data0/geth/chaindata cache=16 handles=16
INFO [01-29|21:21:13] Writing custom genesis block
INFO [01-29|21:21:13] Persisted trie from memory database      nodes=0 size=0.00B
time=358.89µs gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [01-29|21:21:13] Successfully wrote genesis state         database=chaindata
                                          hash=5e1fc7…d790e0
INFO [01-29|21:21:13] Allocated cache and file handles
database=/home/lixu/privatechain/data0/geth/lightchaindata cache=16 handles=16
INFO [01-29|21:21:13] Writing custom genesis block
INFO [01-29|21:21:13] Persisted trie from memory database      nodes=0 size=0.00B
time=2.633µs  gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [01-29|21:21:13] Successfully wrote genesis state
database=lightchaindata                                          hash=5e1fc7…
d790e0
```

初始化成功后，会在数据目录 data0 中生成 geth 和 keystore 两个文件夹，此时目录结构如下：

```
privatechain
├── data0
│   ├── geth
│   │   ├── chaindata
│   │   │   ├── 000001.log
│   │   │   ├── CURRENT
│   │   │   ├── LOCK
│   │   │   ├── LOG
│   │   │   └── MANIFEST-000000
│   │   └── lightchaindata
│   │       ├── 000001.log
│   │       ├── CURRENT
│   │       ├── LOCK
│   │       ├── LOG
│   │       └── MANIFEST-000000
│   └── keystore
└── genesis.json
```

其中 `geth/chaindata` 中存放的是区块数据，`keystore` 中存放的是账户数据。

四、启动私有链节点

```
$ geth --datadir ~/privatechain/data0 --networkid 110  --rpc console
```

上面命令的主体是 `geth console`，表示启动节点并进入交互式控制台，–datadir选项指定使用data0作为数据目录，`--networkid` 选项后面跟一个数字，这里是110，表示指定这个私有链的网络id为110。网络id在连接到其他节点的时候会用到，以太坊公网的网络id是1，为了不与公有链网络冲突，运行私有链节点的时候要指定自己的网络id。

运行上面的命令后，就启动了区块链节点并进入了该节点的控制台：

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

instance: Geth/v1.8.10-unstable-ccc0debb/darwin-amd64/go1.10.2
 modules: admin:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc: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。

五、账号管理

进入以太坊 Javascript Console 后，就可以使用里面的内置对象做一些操作，这些内置对象提供的功能很丰富，比如查看区块和交易、创建账户、挖矿、发送交易、部署智能合约等。Ethereum Javascript API

常用命令有：

- personal.newAccount()：创建账户；
- personal.unlockAccount()：解锁账户；
- eth.accounts：枚举系统中的账户；
- eth.getBalance()：查看账户余额，返回值的单位是 Wei（Wei 是以太坊中最小货币面额单位，类似比特币中的聪，1 ether = 10^18 Wei）；
- eth.blockNumber：列出区块总数；
- eth.getTransaction()：获取交易；
- eth.getBlock()：获取区块；
- miner.start()：开始挖矿；
- miner.stop()：停止挖矿；
- eth.coinbase：挖矿奖励的账户
- web3.fromWei()：Wei 换算成以太币；
- web3.toWei()：以太币换算成 Wei；
- txpool.status：交易池中的状态；
- admin.addPeer()：连接到其他节点；

1. 查看账号

输入 `eth.accounts` 查询系统中的账户：

```
> eth.accounts
[]
```

显示为 `[]`，表示没有账户，接下来使用 `personal.newAccount()` 来创建一个账户：

2. 创建账户

```
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0xb5154c433e9bedabfdb3452c10114d5589b4b62f"
```

`Passphrase` 表示输入密码，`Repeat passphrase` 表示输入确认密码

再次创建一个账户

```
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0xf4e122e28c5c6a84a20c1321147453cd46182464"
```

查看刚刚创建的用户：

```
> eth.accounts
["0xb5154c433e9bedabfdb3452c10114d5589b4b62f",
"0xf4e122e28c5c6a84a20c1321147453cd46182464"]
```

账户默认会保存在数据目录的 `data0/keystore` 文件夹中。可以查看其中的文件

```
{
    "address": "b5154c433e9bedabfdb3452c10114d5589b4b62f",
    "crypto": {
        "cipher": "aes-128-ctr",
        "ciphertext":
"10206abe254d60e220ca788043bcd9a62c5d080f7310621c8f492237476dbd52",
        "cipherparams": {
            "iv": "ca32c24a39f8af68d400a0351755ac9b"
        },
        "kdf": "scrypt",
        "kdfparams": {
            "dklen": 32,
            "n": 262144,
            "p": 1,
            "r": 8,
            "salt":
"3c2070da854a2a45ba0c770d78bd92207e64e3bf247ca47ad834c098b3b38478"
        },
        "mac": "a76a269824d16343a07893dc42ec14cbf477311968b1abc6927a088c39d750d7"
    },
    "id": "6419c0bd-be7e-4f78-a500-e2d345f6fd3f",
    "version": 3
}
```

3. 查看账户余额

通过 `eth.getBalance()` 可以查看账户余额

```
> eth.getBalance(eth.accounts[0])
0
> eth.getBalance(eth.accounts[1])
0
```

目前两个账户的以太币余额都是0，要使账户有余额，可以从其他账户转账过来，或者通过挖矿来获得以太币奖励。

六、 启动和停止挖矿

1. 启动挖矿

通过 `miner.start()` 启动挖矿

```
> miner.start()
```

其中 start 的参数表示挖矿使用的线程数。第一次启动挖矿会先生成挖矿所需的 DAG 文件，这个过程有点慢，如下：

```
> miner.start()
INFO [08-11|13:32:23.819] Updated mining threads                   threads=0
INFO [08-11|13:32:23.819] Transaction pool price threshold updated
price=18000000000
null
> INFO [08-11|13:32:23.819] Commit new empty mining work             number=1
uncles=0
INFO [08-11|13:32:23.819] Commit new full mining work              number=1 txs=0
uncles=0 elapsed=211.408µs
INFO [08-11|13:32:24.999] Generating DAG in progress               epoch=0
percentage=0 elapsed=483.583ms
INFO [08-11|13:32:25.510] Generating DAG in progress               epoch=0
percentage=1 elapsed=994.517ms
INFO [08-11|13:32:26.010] Generating DAG in progress               epoch=0
percentage=2 elapsed=1.493s
```

等进度达到 100% 后，就会开始挖矿，此时屏幕会被挖矿信息刷屏，如下

```
INFO [08-11|13:33:17.719] Generating DAG in progress               epoch=0
percentage=99 elapsed=53.203s
INFO [08-11|13:33:17.720] Generated ethash verification cache      epoch=0
elapsed=53.204s
INFO [08-11|13:33:20.036] Successfully sealed new block            number=1
hash=cd2901…4637f3
INFO [08-11|13:33:20.036] 🔨 mined potential block                  number=1
hash=cd2901…4637f3
INFO [08-11|13:33:20.036] Commit new empty mining work             number=2
uncles=0
INFO [08-11|13:33:20.036] Commit new full mining work              number=2 txs=0
uncles=0 elapsed=215.288µs
INFO [08-11|13:33:20.211] Successfully sealed new block            number=2
hash=62a5e6…14dd79
INFO [08-11|13:33:20.212] 🔨 mined potential block                  number=2
hash=62a5e6…14dd79
INFO [08-11|13:33:20.212] Commit new empty mining work             number=3
uncles=0
```

2. 停止挖矿

不用管刷屏导致的命令不全，命令会正常执行。在 console 中输入：

```
miner.stop()
```

3. 查看矿工账户

挖到一个区块会奖励5个以太币，挖矿所得的奖励会进入矿工的账户，这个账户叫做coinbase，默认情况下coinbase是本地账户中的第一个账户：

```
> eth.coinbase
"0xb5154c433e9bedabfdb3452c10114d5589b4b62f"
```

4. 修改矿工账户

可以通过 `miner.setEtherbase()` 将其他账户设置成 `coinbase` 即可

```
> miner.setEtherbase(eth.accounts[1])
true
> eth.coinbase
"0xf4e122e28c5c6a84a20c1321147453cd46182464"
```

重新启动挖矿，查看 `eth.accounts[1]` 是否可以获得以太币

```
> miner.start(3)

//等待几秒后
> miner.stop()
```

查询账户余额：

```
> eth.getBalance(eth.accounts[0])
55000000000000000000
> eth.getBalance(eth.accounts[1])
35000000000000000000
```

发现账户0 和 账号1 都有以太币，说明 `miner.setEtherbase()` 设置成功。

`getBalance()` 返回值的单位是wei，wei是以太币的最小单位，1个以太币=10的18次方个wei。要查看有多少个以太币，可以用web3.fromWei()将返回值换算成以太币：

```
> web3.fromWei(eth.getBalance(eth.accounts[0]),'ether')
55

> web3.fromWei(eth.getBalance(eth.accounts[1]),'ether')
35
```

七、发送交易

1. 解锁账户

我们从账户0转移10个以太币到账户1,首先要解锁账户 0，才能发起交易：

```
> personal.unlockAccount(eth.accounts[0])
Unlock account 0xb5154c433e9bedabfdb3452c10114d5589b4b62f
Passphrase:
true
```

2. 发起交易

```
> amount = web3.toWei(10,'ether')
"10000000000000000000"
> eth.sendTransaction({from:eth.accounts[0],to:eth.accounts[1],value:amount})

INFO [05-04|10:14:12.015] Submitted transaction
fullhash=0x947b2a4e384b9ba110b05b8e150d6a28fb7c9354f50cc532e494a9b338774619
recipient=0xf4E122E28c5C6A84a20C1321147453cd46182464
"0x947b2a4e384b9ba110b05b8e150d6a28fb7c9354f50cc532e494a9b338774619"
```

3. 账户余额转为ether

查询账户1 的余额：

```
> web3.fromWei(eth.getBalance(eth.accounts[1]),'ether')
35
```

4. 查看交易池交易状态

发现账户余额没有发生改变，此时交易已经提交到区块链，但还未被处理，这可以通过用 `txpool.status` 命令可以看到本地交易池中有一个待确认的交易：

```
> txpool.status
{
  pending: 1,
  queued: 0
}
```

其中有一条pending的交易，pending表示已提交但还未被处理的交易。

5. 指定只挖一个区块

要使交易被处理，必须要挖矿。这里我们启动挖矿，然后等待挖到一个区块之后就停止挖矿：

```
> miner.start(1);admin.sleepBlocks(1);miner.stop();

> web3.fromWei(eth.getBalance(eth.accounts[1]),'ether')
50.000378
```

发现账户收到了账户的钱，还多了5个以太币。其实多出的5个以太币是挖矿奖励。

八、查看交易和区块

1. 查看当前区块总数

```
> eth.blockNumber
19
```

2. 通过区块号查看区块:

```
> eth.getBlock(6)

{
  difficulty: 131328,
  extraData: "0xd98301080c846765746888676f312e31302e328664617277696e",
  gasLimit: 3160033,
  gasUsed: 0,
  hash: "0x024dab0d1bb5da444b783bbc675940bdf92ea2c08c9eb6fb56db07740e47e57d",
  logsBloom:
"0x00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000",
  miner: "0xb5154c433e9bedabfdb3452c10114d5589b4b62f",
  mixHash: "0xc162ad888f90774d2b2acf1400e97a783d2a38b35d3e3498c3610adba54064dc",
  nonce: "0x4898be8dcf2226ec",
  number: 6,
  parentHash: "0x8788ca04121376a99d220621e0bc1a22a3d04d3e642fe0cb42fa8e5f3843cff0",
  receiptsRoot:
"0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  sha3Uncles: "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",
  size: 537,
  stateRoot: "0x56abb05e0ca732749d0ee044e68d044b2a53cbd2570386f6537dd6682d409e73",
  timestamp: 1530670072,
  totalDifficulty: 918144,
  transactions: [],
  transactionsRoot:
"0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  uncles: []
}
```

3. 查看交易

通过交易hash查看交易（hash 值包含在上面交易返回值中）：

```
>
eth.getTransaction("0x947b2a4e384b9ba110b05b8e150d6a28fb7c9354f50cc532e494a9b338774
619")
{
```

```
  blockHash: "0xfbc1a5183feec436f73c181e8f359cf55da6d45a1cb48c82da7c8678eee54a91",
  blockNumber: 19,
  from: "0xb5154c433e9bedabfdb3452c10114d5589b4b62f",
  gas: 90000,
  gasPrice: 18000000000,
  hash: "0x947b2a4e384b9ba110b05b8e150d6a28fb7c9354f50cc532e494a9b338774619",
  input: "0x",
  nonce: 0,
  r: "0xd7c160e09190400d8d3139570f59b722c599f19f130bd317b4603452add36a84",
  s: "0x3358fe8a438bd2645fa5bc4ea3ac55d213559c19493920d345685244b1408c0c",
  to: "0xf4e122e28c5c6a84a20c1321147453cd46182464",
  transactionIndex: 0,
  v: "0x100",
  value: 10000000000000000000
}
```