Ethereum开发安装指南

本文适用对象：

需要在Ethereum上开发程序、完成部署、并提供前端界面的开发人员。本文不涉及BlockChain的技术细节，不讨论miner，不涉及具体语言；本文仅描述私有链的安装、合约向私有链部署、提供界面访问合约。

Ethereum有很多组件，以不同语言开发的多种工具，本文仅描述其中一种：geth，mix，meteor、mist的组合。

Ethereum是在ubuntu上开发的，本文的安装部署环境为ubuntu 14.04 LTS。

最重要的网络资源<https://github.com/ethereum/wiki/wiki>。在这里能找到的各种参数、格式，本文均没有复述。

一．geth

这里描述以geth完成私有链的安装，这个私有链不是一个单机的版本，是可以多台计算机交互的。与[“如何分叉以太坊并变成自己的私链”](http://www.wanbizu.com/baike/201602126543.html)一文相互补充，就可以实现对Ethereum的山寨。

当然不改源代码更简单，本文的安装使用标准的geth，没有修改源代码。

本部分主要的参考文档是

<http://adeduke.com/2015/08/how-to-create-a-private-ethereum-chain/>

<https://github.com/ethereum/go-ethereum/wiki/Setting-up-private-network-or-local-cluster>

1．geth是什么

geth是Ethereum的客户端，Go语言版本，是目前最稳定的客户端；另一个C++版本叫eth。两者实现的功能一致，只是命令、参数不同。

我们只是用客户端，不需要你知道Go，也不需要你知道C++。

geth首先实现了一个p2p的网络，就和BT网络一样，默认使用30303端口与其他peers通讯。

geth实现了挖矿，以及相关的PoW工作。（这事和开发无关，不懂PoW照样开发Ethereum）

geth提供了http接口，以json格式与其他程序交换数据，默认端口8545。后面要提到的mix、meteor都需要这个端口。

BlockChain会改变世界，但从开发者的角度看，geth就是实现了一个分布式数据库。从纯技术的角度看[“私有区块链，共享式数据库一个令人困惑的别名而已”](http://www.8btc.com/private-blockchain)一文说的没错；虽然从业务的角度，我并不 赞同。

geth就是一个可以存数据（账户余额）、可以跑脚本（合约）的数据库。这个数据库是分布式集群的，集群中的节点使用30303端口相互同步；每个节点提供8545端口，作为数据库监听，提供其他程序访问。

2．geth的安装

现在，geth的安装已经简单了，参照 [官方页面](https://github.com/ethereum/go-ethereum/wiki/Building-Ethereum)，输入

bash <(curl -L https://install-geth.ethereum.org)

3．geth的命令行参数

这事需要好好说，很多文档上的参数，并不符合Ethereum开发的需要。

另外，geth仍处于开发阶段，各种命令参数的变化都是突如其来。本文的参数以geth 1.4.3-stable为准（2016年5月版本）。

1）创世块

将下面json存为~/.ethereum/genesis.json，作为创世文件。

{

"nonce": "0xdeadbeefdeadbeef",

"timestamp": "0x0",

"parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",

"extraData": "0x0",

"gasLimit": "0x8000000",

"difficulty": "0x400",

"mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000",

"coinbase": "0x3333333333333333333333333333333333333333",

"alloc": {

}

}

并执行：geth init ~/.ethereum/genesis.json console 2

程序瞬间完成，在.ethereum目录中，已经存在创世块了。

说几点：忘掉 --genesis参数吧，作废了；忘掉 --datadir参数吧，不要改变默认路径；不用在创世文件中给你的账户加余额，挖几分钟矿就有钱了。

我曾经使用默认创世文件，但挖矿不成功，原因不明，可能是难度偏大，我等不下去了。我认为还是自己搞个创世文件的好，创世文件事实上也是一种密码，如果没有这个文件，其他节点是连不上你的网络的。

2）建议执行的命令行参数

geth --verbosity 6 --port 30303 --rpc --rpccorsdomain "\*" --rpcaddr 0.0.0.0 --rpcport 8545 --networkid 6723 --nodiscover console 2

3）--verbosity 6

在自己尝试安装阶段，是建议将信息显示开到最大的。这样可以提供比较多的出错、提示信息。

在geth中，可以使用下面命令调整

web3.debug.verbosity(6);

0为所有不显示，6为所有显示。在以前，这个命令曾叫setVerbosity，admin.verbosity。

4）--port 30303

节点之间同步通讯的端口，除非你在同一台计算机上开多个geth，否则不建议修改。

geth会报upnp30303的错误，不用管它。不知道upnp？自行找网络工程师询问。

5）--rpc --rpccorsdomain "\*" --rpcaddr 0.0.0.0 --rpcport 8545

这些是一组，它的功能，你可以理解为，为数据库打开监听端口。

--rpc必须写，不然其他的参数都不起作用；

--rpcaddr 0.0.0.0，必须有，不然监听开在127.0.0.1上，只有本机能访问。

6）--networkid 6723 --nodiscover

搭建的是私有链，所以自己选一个id吧，而且不需要发现主链节点。

在整个建议的命令行参数中，在你深刻理解各个参数之前，只有这个networkid的值可以改变。当然一个集群中的各个节点，networkid必须是一致的。

4．挖矿

安装好geth后，到目前运行了两句命令

geth init ~/.ethereum/genesis.json console 2

geth --verbosity 6 --port 30303 --rpc --rpccorsdomain "\*" --rpcaddr 0.0.0.0 --rpcport 8545 --networkid 6723 --nodiscover console 2

你应该进入“>”提示符了。下面的命令都是在“>”提示符下的。

1）账户

你应当知道，挖到矿后，会给矿工增加Eth。所以矿工必须有账户。

personal.newAccount("mypassword");

personal.newAccount("mypassword");

personal.unlockAccount(eth.accounts[0], "mypassword");

前两句建了两个账户，后一句是解锁第一个账户。任何从账户花钱的操作，都需要解锁，如转账、部署。

特别重要：这里的mypassword是密码，你可以改为自己的密码，但是绝对不能忘掉。如果忘掉就无法解锁账户，无法从账户出钱，这个账户上所有的钱就没用了。

在newAccount之后，~/.ethereum/keystore目录中会出现账户文件。也必须保存好。

用现实生活中的例子来描述一下：newAccount之后，出现的一串数字，是你的银行卡号。任何人只要知道这个号，就可以给你存钱。

Keystore目录中的文件，是你的银行卡，而mypassword是你的密码。你只有同时拥有这两个，才能从账户中取钱、花钱。

唯一不同的是，这里没有密码恢复的选项，如果忘了密码，这张卡以及卡里的钱就没有了。

其他一些相关的命令包括：

列出所有账户：eth.accounts

指定矿工账户：默认是eth.accounts[0]，可以修改：miner.setEtherbase(eth.accounts[0]);

2）挖矿

命令很简单：miner.start();

以及：miner.stop();

说明几点：第一挖矿机需要4G以上内存，不然，也没什么，就是时间长些。

第一次miner.start()后，会建立DAG，几分钟时间吧。此后不会再建。但以后start()后，还是会有相当长的等待时间，从几秒到十几分钟，你可以看一下硬盘读取速度，从一开始的10M+每秒，逐步下降，降到0以后数秒后开始正式挖矿。

3）余额查看

开始挖矿后，账户上就有Eth了，如果嫌显示太多，可以用web3.debug.verbosity(0);关闭显示。

web3.fromWei(eth.getBalance(eth.accounts[0]), "ether");

4）转账

eth.sendTransaction({from: eth.accounts[0], to: eth.accounts[1], value: web3.toWei(1, "ether")});

这里假设你的eth.accounts[0]上有钱，并已经unlockAccount了。

5．peers

的确，从开发角度看，一个节点与多个节点没什么不同，但是不搞几个节点，怎么体现BlockChain啊。

1）enode

每个节点都有自己的enode，用如下命令显示

admin.nodeInfo.enode

返回结果类似下面 "enode://5a58b99bc3149ad8xxxxxx@[::]:30303"

在命令行用ifconfig找到网卡地址，替换掉中间的[::]，就是这个节点完整的enode了。

如"enode://5a58b99bc3149ad8xxxxxx@192.168.1.1:30303"

2）在另一台电脑上，用同样的创始文件，使用geth命令建立创始块、启动、建立账户（当然这个账户与原电脑上的不同），但不要挖矿。

随后输入

admin.addPeer("enode://5a58b99bc3149ad8xxxxxx@192.168.1.1:30303");

如果顺利，新电脑就会开始同步数据。

如果有问题，两边控制台的显示会有足够的提示信息。这就是verbosity(6);的好处。

另外说一句：如果你是clone、bridge后，发现有奇怪的连接问题；提示一句：MAC冲突。（懂者自知，费了我一下午时间）。

还有，MAC改变后，会无法unlockAccount。没人跟我说过啊，难道网卡坏了，所有的钱也就丢了？在确认这个问题之前，请保存好运行geth电脑的MAC地址。

3）当然，如果你非要在同一台电脑上运行另一个节点，也行。改变port，rpcport以及ipcpath（对这个参数我上面没提，因为我不建议这么做）。

二．mix-ide

上面是对geth的描述，从开发的角度了解这些就够了。

1．mix-ide是什么

geth建立了EVM环境，可以在上面运行的合约语言，类似于伪汇编。没人会用汇编写合约的，Solidity（目前最流行），Serpent，LLL就是几种高级语言。Solidity经过solc编译后，成为汇编语言。通过部署，将这种汇编语言放到BlockChain上。

truffle是这样一种命令行工具，有编译、部署等功能。还有很多工具提供这些功能，但目前推荐mix-ide，因为他除了提供编译Solidity语言、部署合约外，还提供较完善的调试环境，用以调试合约语言。

按照前面的类推，geth是数据库，mix是PL/SQL，合约语言就是运行在数据库上的存储过程，这种存储过程使用Solidity语言编写。

2．mix的安装

参照[官方网页](http://www.ethdocs.org/en/latest/ethereum-clients/cpp-ethereum/installing-binaries/linux-ubuntu-ppa.html)，在命令行输入

sudo add-apt-repository ppa:ethereum/ethereum-qt

sudo add-apt-repository ppa:ethereum/ethereum

sudo apt-get update

sudo apt-get install cpp-ethereum

就是时间稍长，半个小时吧，很简单了。半年前我装的时候，花了两天。

3．合约程序

只是练习安装，不用编，就用new以后自动生成的Sample就行。

4．部署

使用mix没什么好说的，选deploy to network。这里也不说solidity语言。就讲关于部署的几个关键点。

1）部署的时候，看到rpcport了吧，这个都会解决。

2）部署前，需在geth上将部署账户unlockAccount，不然你就傻等吧。部署是要花费Eth的，所以必须解锁。

3）部署时，要有一定的耐心，在单机的testrpc上，部署也许很快；在按照本文安装的private chain上，部署也是需要几分钟的。

4．打包

最后生成packet/www里的文件，待用。

三．meteor

与传统web开发一样，数据库、存储过程搞好了以后，就是找个中间件了。

我是把nodejs与meteor都搞通了，比较后，觉得meteor更好，这里就讲meteor了。

另外mist很好玩，按照[这篇](http://ethfans.org/topics/118)走一遍也很有趣。

先说点其他的。Ethereum分四个阶段Frontier、Homestead、Metropolis、Serenity。

经过9次PoC（原型证明）Frontier于2015年9月（好像是的，不记得了）发布，主要完成以geth为代表的客户端。

Homestead于2016年2月发布，3月14日完成硬分叉，开始了与Frontier不同的挖矿旅程，主要完成各种Dapp开发工具。

Metropolis主要完成以mist为代表的Ethereum浏览器。在mix上部署完成后生成的packet，可以直接给mist使用。但目前，不能指望一般用户会使用mist，他们只会用普通的IE、chrome。所以还需要传统的中间件。

Serenity还处在PoC阶段（2016年5月），这个阶段同前三个比，会有比较大的改变。

1．meteor干什么

本节标题不是“是什么”，而是“干什么”。因为meteor本身和Ethereum没有关系。

如果不用meteor这类的中间件，也行。

geth提供了JSON RPC接口，就是8545的接口，你按照这个同样可以实现所有的功能。而web3.js就是将这种接口进行了封装，提供了类似javascript语言的接口。并按nodejs、meteor的格式提供了调用包。

2．安装

基本上就是按照[官网教程](https://github.com/ethereum/wiki/wiki/Dapp-using-Meteor)走的，说明一下是，curl https://install.meteor.com/ | sh总不成功。我是用[github](https://github.com/meteor/meteor)上的Slow Start (for developers)，用

git clone git://github.com/meteor/meteor.git

cd meteor

meteor

完成安装的。

3．生成项目

生成空项目：meteor create myDapp

cd myDapp

添加包：meteor add ethereum:web3

其他的ethereum:dapp-styles、ethereum:tools、ethereum:elements、ethereum:accounts、ethereum:blocks、frozeman:template-var、frozeman:persistent-minimongo2一并装了吧。在搞懂之前，多装点东西无所谓。

4．加入rpcport

直接在myDapp/client/lib目录下增加init.js。内容为

if(typeof web3 === 'undefined')

web3 = new Web3(new Web3.providers.HttpProvider('http://localhost:8545'));

不需要在其他地方引用init.js。

5．最后就是程序了。

我是把meteor生成的程序稍改了一下，实际就是把packet/www里的东西加了进来。

用你自己packet/www里面的东西对应一下，就可以访问到你自己的区块链合约了。

如果你用了和我一样的示例，那么差别只是那个address的值了。

**main.html**

<head>

<title>myDapp</title>

</head>

<body>

<h1>Welcome to Meteor!</h1>

{{> hello}}

{{> getv}}

{{> info}}

</body>

<template name="hello">

<button>Click Me</button>

<p>You've pressed the button {{counter}} times.</p>

<p>Now block number is {{currentBlock}} .</p>

</template>

<template name="getv">

<button>Get</button>

<p>Rate value is {{currVal}} .</p>

</template>

<template name="info">

<h2>Learn Meteor!</h2>

<ul>

<li><a href="https://www.meteor.com/try">Do the Tutorial</a></li>

<li><a href="http://guide.meteor.com">Follow the Guide</a></li>

<li><a href="https://docs.meteor.com">Read the Docs</a></li>

<li><a href="https://forums.meteor.com">Discussions</a></li>

</ul>

</template>

**main.js**

import { Template } from 'meteor/templating';

import { ReactiveVar } from 'meteor/reactive-var';

import './main.html';

contracts = {};

ctrAddresses = {};

contracts["Sample"] = {

interface: [{"constant":false,"inputs":[{"name":"v","type":"uint256"}],"name":"set","outputs":[],"type":"function"},{"constant":true,"inputs":[],"name":"get","outputs":[{"name":"","type":"uint256"}],"type":"function"},{"inputs":[{"name":"v","type":"uint256"}],"type":"constructor"}],

address: "0xc702942ec216c6a701ffa772c0a9f30580619228"

};

contracts["Sample"].contractClass = web3.eth.contract(contracts["Sample"].interface);

contracts["Sample"].contract = contracts["Sample"].contractClass.at(contracts["Sample"].address);

if (!ctrAddresses["Sample"]){ ctrAddresses["Sample"] = [] }ctrAddresses["Sample"].push("0xc702942ec216c6a701ffa772c0a9f30580619228")

Template.hello.onCreated(function helloOnCreated() {

this.counter = new ReactiveVar(0);

EthBlocks.init();

});

Template.getv.onCreated(function getvOnCreated() {

this.currVal = new ReactiveVar(0);

});

Template.hello.helpers({

counter() {

return Template.instance().counter.get();

},

currentBlock: function(){

return EthBlocks.latest.number;

}

});

Template.getv.helpers({

currVal() {

return contracts['Sample'].contract.get();

},

});

Template.hello.events({

'click button'(event, instance) {

// increment the counter when button is clicked

instance.counter.set(instance.counter.get() + 1);

},

});

Template.getv.events({

'click button'(event, instance) {

instance.currVal.set(contracts['Sample'].contract.get());

},

});

6．运行

meteor

然后打开浏览器，<http://localhost:3000>即可

四．总结：

本文描述了Ethereum开发，所需要的环境安装，工具安装，以及各个工具之间的关联关系。以及我所遇到的一系列的坑。

希望能够加快大家安装的速度。

完成这些工作后，下面就是学习Solidity，meteor了。