[英文文档](http://truffle.readthedocs.io/en/latest/)

环境准备：

     系统环境：Linux 4.2.0-27-generic ~14.04.1-Ubuntu  x86\_64 x86\_64 x86\_64 GNU/Linux

     node.js  （v6.2.0）

     npm （3.8.9）

     pm2 （1.1.3）

     testRPC（v2.0.7）

     Truffle （v1.0.3）

读和写数据：

  写数据（Transaction/处理事务）：需要付费，将会改变网络节点的状态，不会立即执行，无返回值（只有一个事物ID标志）

  读数据（Calls/调用）：免费，不会改变网络节点的状态，会立即执行，有返回值

1. **truffle相关命令讲解**

truffle 命令格式：

     $ truffle [command] [options]

1>$ truffle build [ -e 环境类型 ]  建立一个开发版本的程序

     版本类型有四种： development（开发环境，也是默认类型）  staging（测试环境：可共享测试的blockchina，可多人使用）  production（正式环境，会连接到公有网络）  test（测试环境）

2>$ truffle console [-e 环境类型] [ --verbose-rpc “path” ]  用于进入truffle控制台

     [ --verbose-rpc “path” ] turffle与rpc通讯日志位置

3>$ truffle compile [ --compile-all ]   编译合约

     默认情况下只会去编译修改过的合约，当加入 --compile-all 参数时，表示编译所有合约

4>$ truffle create:contract [合约名称（eg:MyContract）] 辅助构建一份新合约

     合约名称必须满足“驼峰命名”格式

5>$ truffle create:test [合约名称（eg:MyTest）]  辅助创建一个测试合约

6>$ truffle deploy [ -e 环境类型 ] [ --compile-all ] [ --verbose-rpc “path”]  发布合同到网络

     [ --compile-all ] 指定编译当前环境下的所有合同

     [ --verbose-rpc “path” ] turffle与rpc通讯日志位置

7>$ truffle dist [ -e 环境类型 ]   编译一个带版本好的合同 会创建 ./environments/<name>/dist 目录

8>$ truffle exec [ -e 环境类型] /path/to/my/script.js   执行某个脚本

    $ truffle-exec <path/to/file.js>  与上一个命令功能一直

    上述两个方法有个局限，脚本必须在结束时调用 process.exit() 方法，要不然此脚本就不会结束

9>$ truffle init    初始化一个默认的工程

      初始化一个全新的应用程序在当前工作目录。将添加默认合同,测试和前端配置。

10>$ truffle init:config  和 9 中的init 命令很像，只是这个命令只会初始化config目录

11>$ truffle init:contracts  和 9 中的init 命令很像，只是这个命令只会初始化contracts  目录

12>$ truffle init:tests  和 9 中的init 命令很像，只是这个命令只会初始化tests目录

13>$ truffle list  列出所有的命令并退出，类似于 --help

14>$ truffle resolve ./path/to/contract/file.sol  解决可靠性文件内的所有依赖项并打印结果

15>$ truffle serve [ -e 环境类型 ] [ -p port] 重新构建合约并将合约发布到网络

     [ -p port] 指网络端的端口（默认为8545）

16>$ truffle test [ -e 环境类型 ] [ --compile-all ] [ --verbose-rpc “path” ] [/path/to/test/file] 执行测试文件下的文件（.js , .es , .es6 , .jsx）

      [/path/to/test/file] 需要执行的文件（注：可被执行的文件只有四种：.js , .es , .es6 , .jsx q其他文件都不能被执行 ）

17>$ truffle version  显示版本号并退出

18>$ truffle watch  监控合约,应用程序和配置文件的变化，当其中一个或多个变化时，进行重新编译和发布

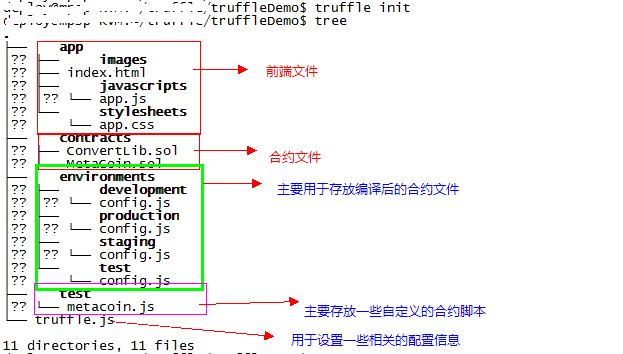
**二.通过truffle 创建一个合约**

1>创建一个目录作为工程目录：$ mkdir truffleDemo

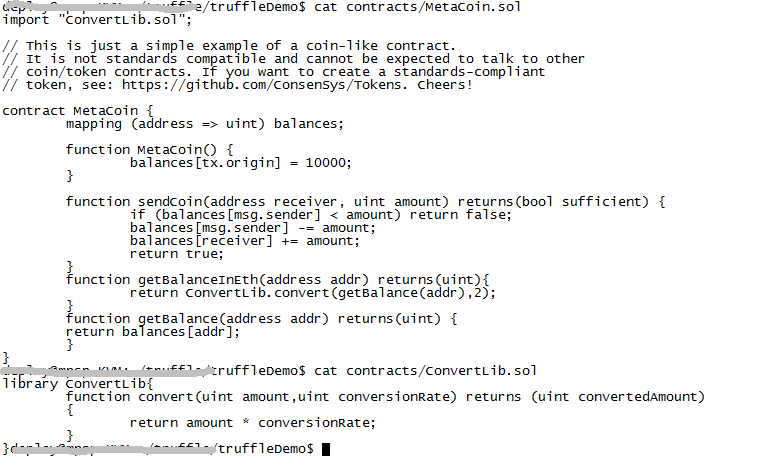
2>$ cd truffleDemo

3>创建一个完整的工程目录：$ truffle init

 创建完整 之后的目录结构如下：



4>我们可以看到，文件中已经存在了一个合约文件：MetaCoin.sol ，现在我们来看看此合约文件内容：



我们可以看到，合约中

 1.通过 import 引入了一个.sol 汇率转换合约文件

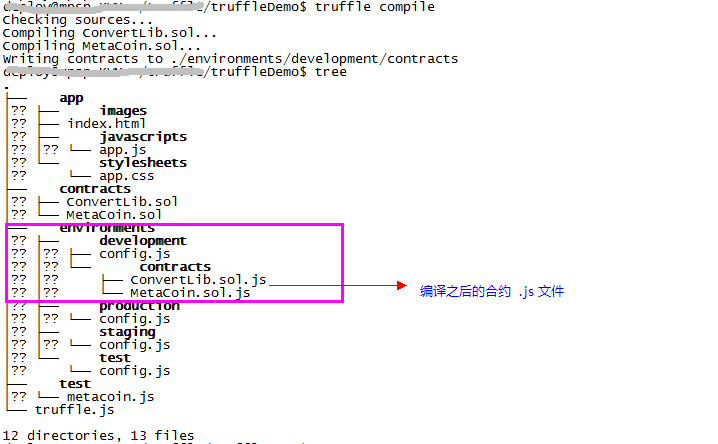
 2.一个账户用于表示账户余额的变量 balance

 3.一个构造方法MetaCoin（）

 4.一个用于发送货币的方法 sendCoin（）

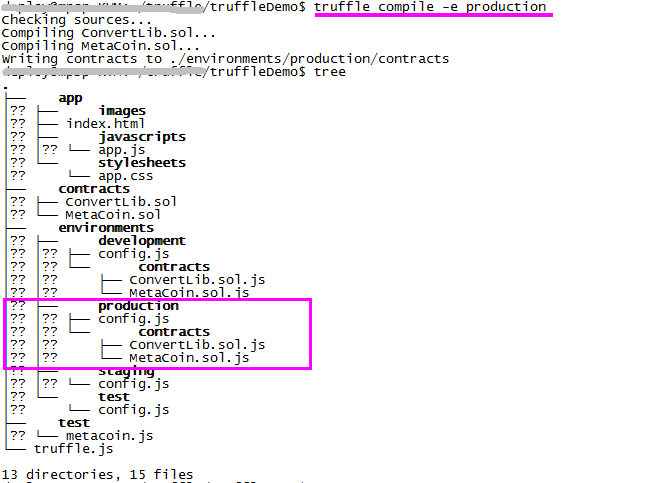
 5.一个获取账户余额的方法 getBalance（）

5>现在我们对合约进行编译：$ truffle compile [ -e 环境类型 ]

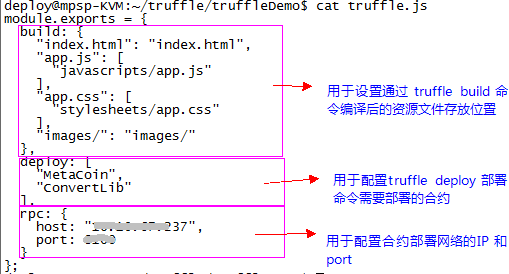


  默认情况下，编译后的合约是放在开发环境目录下的，就如上文命令中讲解的，我们可以通过 -e 参数设置编译的环境

  如：$ truffle compile -e production  编译后的合约被放在了正式环境 /production 目录下



6>现在合约编译完成了，我们需要修改 truffle 的主配置文件 truffle.js ，配置一些部署此合约需要的配置项



注意：

  1.合约的名称是大小写敏感的，所以一定要确认合约名称的正确性

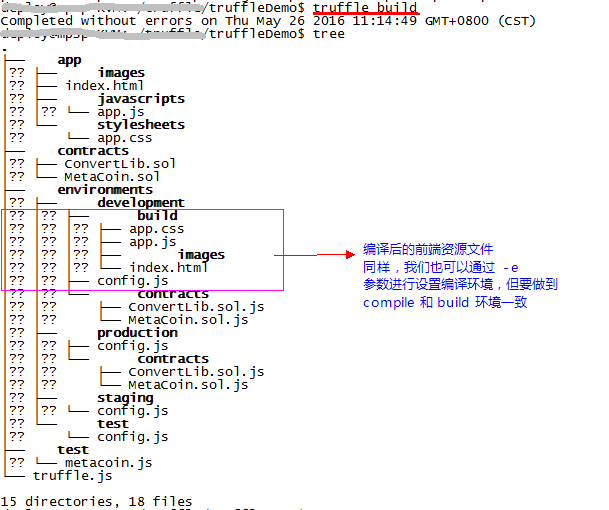
  2.在 rpc 配置项中，除了 host 和 port 参数之外，还可以指定 gas（部署合约所需费用的上限，默认值为3141592），gasPrice（部署合约需要的费用，默认值为100000000000，即100香农），from （指定部署合约所用的账户， 默认为第一个可用的Ethereum客户端提供的账户。）

  3.在此配置文件中，如果你想要在部署完成之后指定指定的.js 文件，可以添加一个 after\_deploy 配置项，此配置项是一个数组配置项，如上图中的deploy项，在里面可以指定一个或多个需要执行的 .js 文件路径

  4.在此配置文件中还可以添加 mocha 配置项， 此项是MochaJS测试框架的配置选项。

**重点注意：truffle.js 文件的配置信息也可以配置在各个环境中的 config.js 文件中，起到一一对应的效果**

7>现在我们编译前端资源环境：$ truffle build [ -e 环境类型 ]



8>现在我们开始部署我们的合同：$ truffle deploy [ -e 环境类型 ]

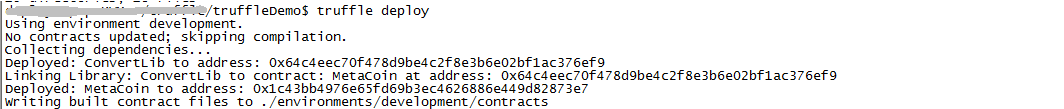
 注意：在部署合同之前，一定要解锁前面在配置文件truffle.js 中配置的 from 对应的账户，如果没有配置，则默认为第一个账户

 我在此处链接的是本地搭建的私链，因此只需要解锁本地私链账户即可，如果没有私链，也可以通过使用 testRpc 进行测试

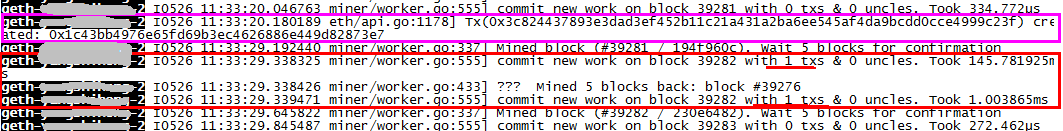
解锁命令：在geth 控制台中执行$>personal.unlockAccount(eth.accounts[0])或 $>personal.unlockAccount("from中配置的账户地址")

   命令解释：personal.unlockAccount（"账户地址 "，"密码"，账户保持解锁状态的时间）

账户被解锁之后，我们开始部署合约：



geth 后台输出的信息中我们可以看到我们部署合约的处理事务，可以看到，这个事务是被 第 39282 个块进行处理的，同样我们可以通过 debug.printBlock(块编号) 命令进行调试块信息：

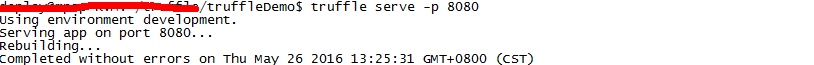


同样的我们也可以在监控页面看到事务的处理过程及费用的花费信息：

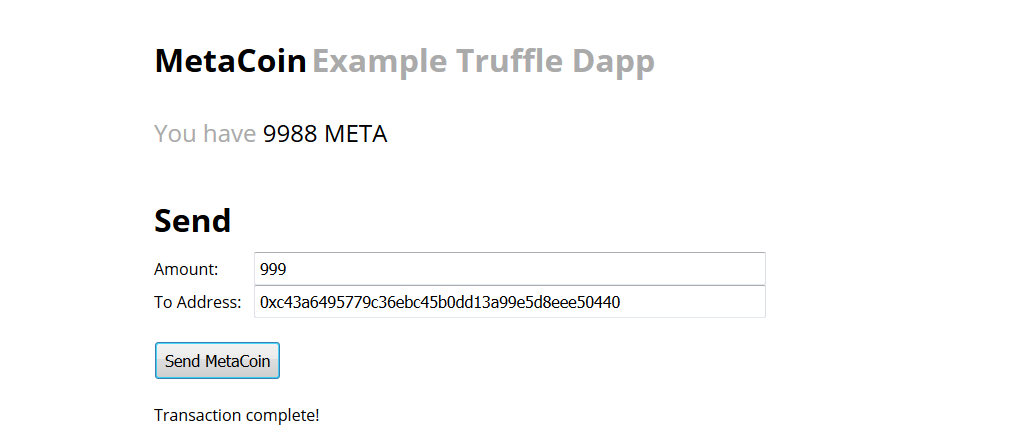




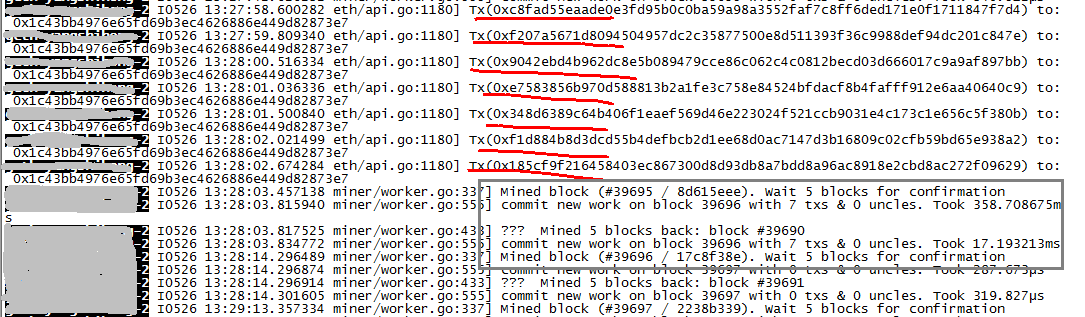
9>现在我们可以启动全段页面对合约进行操作：$truffle serve -p 8080



通过 http://{ip}:8080 访问全段页面



通过向 地址 ：0xc43a6495779c36ebc45b0dd13a99e5d8eee50440 转账 999 ，可已在geth 后台监控到事物信息：



监控页面也会监控到相应的信息：



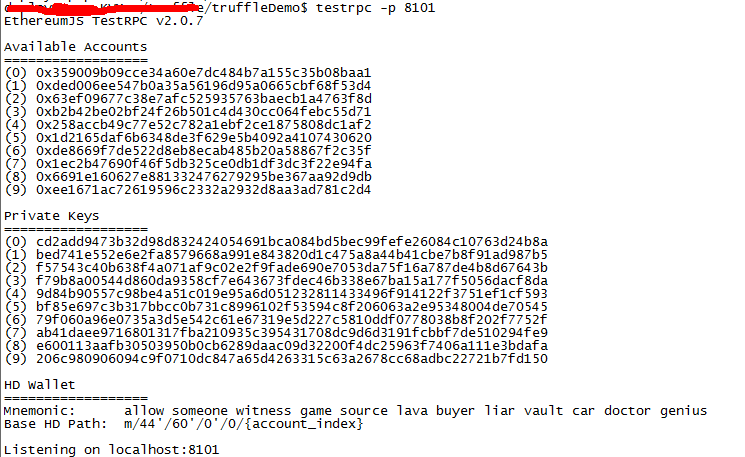
**三：通过testRpc 进行测试部署合约**

如果只是做测试用，testRPC 是一个不错的选择，其相当于一个私链测试环境，在启动 testRPC 之后，会随机生成 10 个 账户地址和钱包地址供开发测试使用

1>先安装 testRPC：$ npm install -g ethereumjs-testrpc

2>启动 testRPC：$ testrpc -p 8101

     testRPC 的命令参数请参考下文附录中的testRPC网站



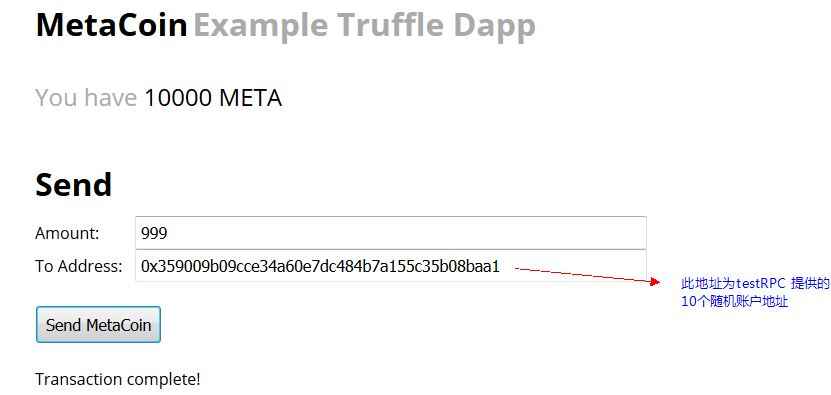
3>修改合约配置文件 truffle.js



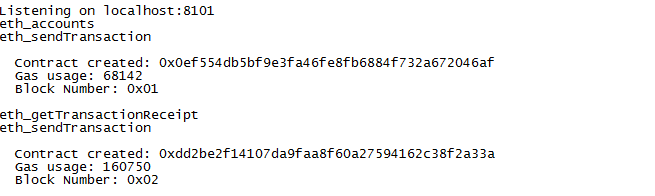
4>部署合约并启动前端服务：



5>通过前端页面进行转账操作，会在testRpc 后台输出相应的信息：



testRPC 会输出相应的监控信息：



附：可能需要了解的信息地址

nodejs , npm ,pm2 的安装可以参考 <http://ethfans.org/wikis/星火节点计划-Ubuntu-接入文档>

testRpc相关信息地址：

<https://github.com/ethereumjs/testrpc>

搭建本地私链参考信息：

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

mocha相关信息地址：

<http://mochajs.org/>

<https://github.com/mochajs/mocha/wiki/Using-mocha-programmatically>