

EOS DAWN 3.0 安装及智能合约初体验



RhainL (/u/d5776bb33ff5) [+ 关注](#)

2018.04.09 22:12 字数 1615 阅读 108 评论 1 喜欢 4

(/u/d5776bb33ff5)

EOS DAWN 3.0 已经正式发布了，这次终于可以在mac上跑起来了，之前的版本，各种折腾都没有真正跑起来，最多只是出了块，但是智能合约没有成功执行。下面主要介绍Mac下启动EOS DAWN3.0 以及运行Currency智能合约。

下载源码

命令行执行

```
git clone https://github.com/EOSIO/eos --recursive
```

构建EOS

下载完源码后，我们可以使用EOS提供的自动构建脚本来构建EOS。

```
cd eos
./eosio_build.sh
```

在使用自动构建的时候，不一定会构建成功，我就是碰到了这个问题。EOS 需要先安装一些依赖，自动脚本理论上可以自动帮你安装这些依赖，但是由于环境的不同，导致有些依赖并不能安装成功，那就需要自动手动去把依赖安装好了。这里说下我遇到的问题

1. MongoDB 自动脚本安装失败，这个是我手动执行 `brew install mongodb` 安装成功。
2. doxygen 自动脚本安装失败。这个问题是brew 下载doxygen的安装程序失败了，但是在浏览器上却是可以下载下来的。解决办法是，先看下 `/usr/local/Library/Formula` 目录中是否存在 `doxygen.rb` 文件，一般在 `brew install doxygen` 的时候会下载下来。如果 `doxygen.rb` 文件存在，根据 `brew install doxygen` 安装时显示的日志可以找到文件的下载路径 `https://ftp.stack.nl/pub/users/dimitri/doxygen-1.8.14.src.tar.gz` 把这个文件下载下来，放到 `~/Library/Caches/Homebrew` 目录中。在执行 `brew install doxygen`。这样doxygen应该就可以安装成功了，后面再执行 `./eosio_build.sh` 就能成功构建EOS了。

当然也可以不使用自动构建脚本来构建EOS，参照官方文档手动吧所有依赖都安装了。这里就不多做介绍了。

启动单节点测试网络

成功安装EOS后，我们可以在本地启动一个单节点的测试网络。

可以直接通过一个命令启动一个单节点网络

```
nodeos -e -p eosio --plugin eosio::wallet_api_plugin --plugin eosio::chain_api_plugin --plugin eosio::acc
```

这样的坏处就是，以后每次启动都需要手动输入这么长的字符串，操作会比较麻烦。

EOS为我们提供了配置文件可以省去后面的配置选项。默认EOS的配置放置在 `~/Library/Application Support/eosio/nodeos/config` 目录中。可以通过 `--config-dir` 这个参数指定另外的目录当做放置配置文件的目录。注意，这个目录一开始是不存在的，可以在 `build/programs/nodeos` 目录中执行下 `./nodeos`，然后立即用 `Ctrl + C` 关闭。EOS会自动生成这个目录，同时生成`config.ini`和`genesis.json`两个文件。

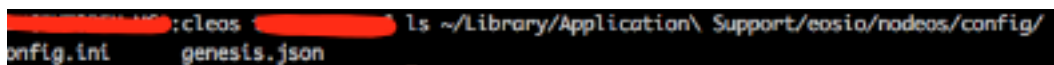


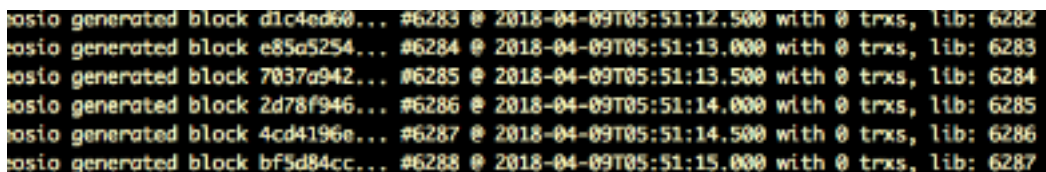
image.png

接下来我们需要修改下`config.ini`文件，直接把下面这段配置复制到`config.ini`文件后面即可。

```
# Enable production on a stale chain, since a single-node test chain is pretty much always stale
enable-stale-production = true
# Enable block production with the testnet producers
producer-name = eosio
# Load the block producer plugin, so you can produce blocks
plugin = eosio::producer_plugin
# Wallet plugin
plugin = eosio::wallet_api_plugin
# As well as API and HTTP plugins
plugin = eosio::chain_api_plugin
plugin = eosio::http_plugin
# This will be used by the validation step below, to view account history
plugin = eosio::account_history_api_plugin
```

官方文档显示`genesis.json`也需要修改，其实是不需要修改的，直接使用默认的就行。

最后就是在 `eos/build/program/nodeos/`目录执行 `./nodeos` 就可以成功启动EOS了。成功启动后可以看到已经在出块了。



运行Currency智能合约

EOS默认已经提供了一些智能合约的样例。我们直接拿来使用就可以了。

创建一个钱包

每个一个智能合约都需要一个关联的账户，账户需要使用钱包来创建，所以我们在启动EOS的时候需要加载钱包插件来创建账户，在之前的配置文件中，我们已经加载了钱包插件，所以这里我们就不需要做什么额外的操作了。

使用 `cleos` 的 `wallet create` 命令来创建一个钱包：

```
cd ~/eos/build/programs/cleos/  
./cleos wallet create
```

这就会创建一个默认的钱包，同时会输出一个密码，自己保存好，以备后面使用。

加载 Bios 智能合约

设置 `eosio.bios` 合约为系统默认合约。这个合约可以让我们直接控制其他账户的资源分配和一些私有api调用。

```
$ ./cleos set contract eosio ../../contracts/eosio.bios -p eosio
```

创建 currency 合约账户

为currency 合约生成一个currency账户，需要两个 公钥、私钥对，一个作为 `public-OwnerKey` 一个作为 `public-ActiveKey`。

```
cd ~/eos/build/programs/cleos/  
./cleos create key # OwnerKey  
./cleos create key # ActiveKey
```

这将会输出两个 公私钥对，像下面这样：

```
Private key: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
Public key: EOSXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

保存好这两个公私钥对，后面会使用到。

把生成好的两个私钥导入钱包：

```
./cleos wallet import <private-OwnerKey>
./cleos wallet import <private-ActiveKey>
```

使用 `cleos create account` 命令来生成 currency 账户。使用 `eosio` 来认证 currency 账户。eosio 是我们在 `genesis.json` 文件中指定的 producer 账户。上面生成的公钥给 currency 账户当作 OwnerKey 和 ActiveKey。

```
./cleos create account eosio currency <public-OwnerKey> <public-ActiveKey>
```

执行后会有结果返回，像这样：

```
executed transaction: fe5c9db1b5173dd4bd1ed79c23056104427ab62b0086cf117175abb322532d93 346 bytes 101544
#          eosio <= eosio::newaccount          {"creator":"eosio","name":"currency","owner":{"threshold"
```

验证账户是否创建成功：

```
./cleos get account currency
```

一切正常的话，会输出类似下面的内容：

```
{
  "account_name": "currency",
  "permissions": [{
    "perm_name": "active",
    "parent": "owner",
    "required_auth": {
      "threshold": 1,
      "keys": [{
        "key": "EOS8kjeKVzFfqYyqcG8EnRLvMyLjJ7nmSM8p7QqDazGnjMEtQdldp",
        "weight": 1
      }]
    },
    "accounts": []
  }],
  {
    "perm_name": "owner",
    "parent": "",
    "required_auth": {
      "threshold": 1,
      "keys": [{
        "key": "EOS6eRfSRYNcrsLmLMomWbBk317gz2TcBqArL7JwaqvaYkWYALe73",
        "weight": 1
      }]
    },
    "accounts": []
  }
]
}
```

上传 currency 合约到 区块链上

在上传之前，我们可以验证下区块链上是否已经有currency合约：

```
./cleos get code currency
code hash: 0000000000000000000000000000000000000000000000000000000000000000
```

一串 0 表示区块链上还没有currency合约。

使用 currency 账户上传 currency合约：

```
./cleos set contract currency ../../contracts/currency
```

执行正常会返回一个 `transition_id` 的json字符串。

同样我们可以验证合约是否上传成功：

```
./cleos get code currency
```

如果返回像下面的内容，则表示合约上传成功：

```
code hash: 9b9db1a7940503a88535517049e64467a6e8f4e9e03af15e9968ec89dd794975
```

在使用currency合约之前，我们需要先创建在发行这个currency：

```
./cleos push action currency create '{"issuer":"currency","maximum_supply":"1000000.0000 CUR","can_freeze":true}' --permission eosio:currency
./cleos push action currency issue '{"to":"currency","quantity":"1000.0000 CUR","memo":""}' --permission eosio:currency
```

接下来验证下currency账户的初始余额：

```
./cleos get table currency currency accounts
{
  "rows": [{
    "balance": "1000.0000 CUR",
    "frozen": 0,
    "whitelist": 1
  }],
  "more": false
}
```

可以看到currency账户有了 1000的CUR。

使用currency合约转账

使用 currency合约的 transfer action 从currency账户转账给eosio账户：

```
./cleos push action currency transfer '{"from":"currency","to":"eosio","quantity":"20.0000 CUR","memo":""}' --permission eosio:currency
```

如果执行成功会有类似下面的输出：

```
executed transaction: de83ee65f983be89bebd2fc5d5ba066acaadcdebdbfc15f8f1221b98f76551ea 271 bytes 109135
#      currency <= currency::transfer      {"from":"currency","to":"eosio","quantity":"20.0000 CUR",
>> transfer
#      eosio <= currency::transfer      {"from":"currency","to":"eosio","quantity":"20.0000 CUR",
```

检查currency账户余额

先看下 eosio的余额情况：

```
./cleos get table currency eosio accounts
{
  "rows": [{
    "balance": "20.0000 CUR",
    "frozen": 0,
    "whitelist": 1
  }],
  "more": false
}
```

可以看到eosio 账户已经有 20的CUR了

在看下currency账户的余额：

```
./cleos get table currency currency accounts
{
  "rows": [{
    "balance": "980.0000 CUR",
    "frozen": 0,
    "whitelist": 1
  }],
  "more": false
}
```

也可以看到currency 账户初始有 1000的CUR ,转了20给 eosio账户，现在还剩余980 CUR。

这样一个简单的currency智能合约就完成了。这个currency合约有点以太坊ERC20 token的意思。操作感觉也比较简单。