让 EOS 跑起来

EOS 编译、测试指南（一）

扫一扫关注欧链小秘书

EOS 团队于 2017 年 7 月 28 日推出了单机测试版，基于此单机版开发者可以完成用户注册、转账等简单功能。OracleChain 团队在第一时间对代码进行了编译和测试，以下将详细介绍如何让 EOS 在自己的本地跑起来。OracleChain 团队使用 Mac 系统进行开发，并使用 Homebrew 进行软件管理。

该指南将从环境准备、获取 EOS 代码、编译 EOS 代码和运行 EOS 四个方面对 EOS 的开发进行讲解，帮助开发者进入 EOS 世界。

EOS 文档：https://eosio.github.io/eos/

EOS 代码：https://github.com/EOSIO/eos

第一步：环境准备

EOS 是基于 C++ 14 进行开发并使用 CMake 进行编译管理，据 git 上的信息，EOS 开发者使用 clang 4.0.0 和 CMake 3.8.0 进行开发编译。

EOS 使用 WebAssembly 对编译和运行智能合约，因此需要使用 WASM 编译器。

除此之外 EOS 还依赖：Boost 1.64，OpenSSL， LLVM 4.0 和 secp256k1-zkp。

基本环境安装

Mac 系统下需先安装 xcode，然后运行：

brew install automake autoconf libtool cmake。

Boost 安装

brew install boost

OpenSSL 安装

brew install openssl

在 Mac 系统已经移除了 openssl，需要手动配置环境变量：

在~/.bash_profile 内添加：

export OPENSSL_ROOT_DIR=/usr/local/Cellar/openssl/1.0.2l
export OPENSSL_INCLUDE_DIR=/usr/local/Cellar/openssl/1.0.2l/includes

其中 1.0.2l 为 openssl 版本号。

随后更新配置文件：

source ~/.bash_profile

安装 secp256k1-zkp

1. git clone https://github.com/cryptonomex/secp256k1-zkp.git
2. cd secp256k1-zkp
3. ./autogen.sh
4. ./configure
5. make
6. sudo make install

安装 LLVM

brew install llvm

随后添加环境变量，在~/.bash_profile 内添加：

```
export PATH="/usr/local/opt/llvm/bin:$PATH"
```

编译 WASM 编译环境

1. mkdir ~/wasm-compiler
2. cd ~/wasm-compiler
3. git clone --depth 1 --single-branch --branch release_40 https://github.com/llvm-mirror/llvm.git
4. cd llvm/tools
5. git clone --depth 1 --single-branch --branch release_40 https://github.com/llvm-mirror/clang.git
6. cd ..
7. mkdir build
8. cd build
9. cmake -G "Unix Makefiles" -DCMAKE_INSTALL_PREFIX=..
   -DLLVM_TARGETS_TO_BUILD=
   -DLLVM_EXPERIMENTAL_TARGETS_TO_BUILD=WebAssembly
   -DCMAKE_BUILD_TYPE=Release ../
10. make -j4 install

至此，准备工作已经完成。

第二步，获取 EOS 代码

EOS 代码使用了三个子模块，包括两个 EOS 自己维护的插件管理模块 AppBase 和区块链结构模块 ChainBase，以及 WASM 模块。

开发者可以通过 git clone https://github.com/eosio/eos --recursive 获取全部代码，或者在获取 EOS 代码后通过 git submodule update --init –recursive 补全子模块。

第三步，编译 EOS 代码

建议使用前面准备的 WASM 编译器对 EOS 进行完整编译。

1. cd eos

2. mkdir build && cd build
3. export WASM_LLVM_CONFIG=~/wasm-compiler/llvm/bin/llvm-config
4. cmake ..
5. cd ..
6. make -j4

其中~/wasm-compiler/llvm/bin/llvm-config 为之前编译的 WASM 编译器地址。开发者可以将 WASM_LLVM_CONFIG=~/wasm-compiler/llvm/bin/llvm-config 添加到.bash_profile 中去。

至此，eos 已经编译完成。

第四步，运行 EOS

在 eos/programs 文件加下，eosd 是单机版的 EOS 节点，会模拟多个账号轮流出块。eosc 通过 REST 访问 eosd，并提供命令行工具。

运行 eosd

首次运行 eosd 下的 eosd 将会报错，并会在 eosd 文件夹下生成 data-dir 文件夹，此时需要对文件夹下的 config.ini 文件进行修改然后再重新运行 eosd。

注释掉原文中的 enable-stale-production = false。

在 config.ini 文件末尾添加

```
# Load the testnet genesis state, which creates some initial block producers with the default key
genesis-json = /path/to/eos/genesis.json
# 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 = inita
producer-name = initb
producer-name = initc
producer-name = initd
```

```
producer-name = inite
producer-name = initf
producer-name = initg
producer-name = inith
producer-name = initi
producer-name = initj
producer-name = initk
producer-name = initl
producer-name = initm
producer-name = initn
producer-name = inito
producer-name = initp
producer-name = initq
producer-name = initr
producer-name = inits
producer-name = initt
producer-name = initu
# Load the block producer plugin, so we can produce blocks
plugin = eos::producer_plugin
plugin = eos::chain_api_plugin
```

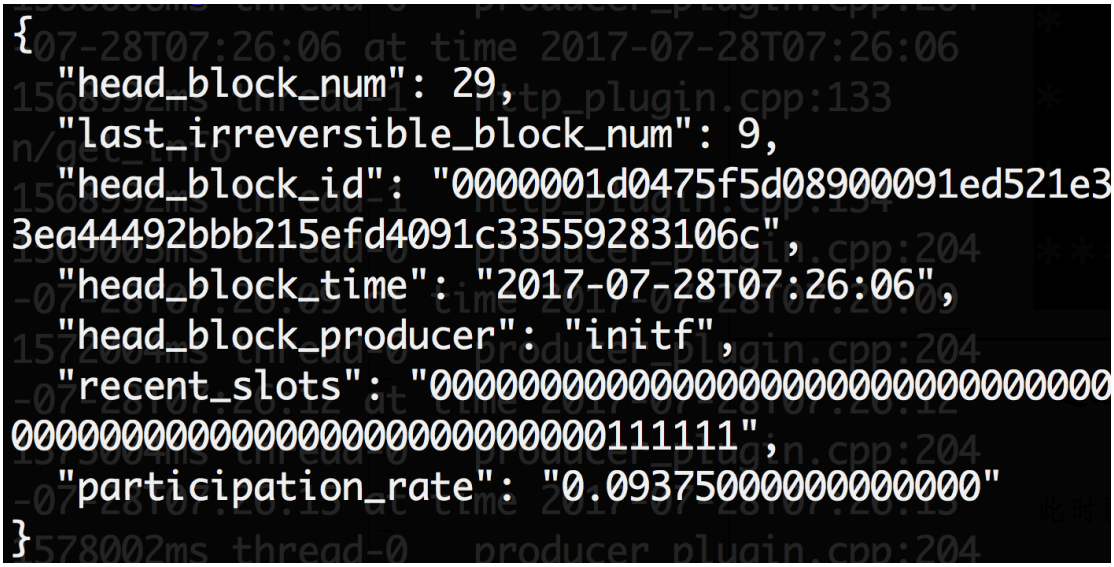
其中/path/to/eos/genesis.json 是 genesis.json 文件的全地址，在 eos 文件夹下。

随后再次运行 eosd，将启动 EOS。

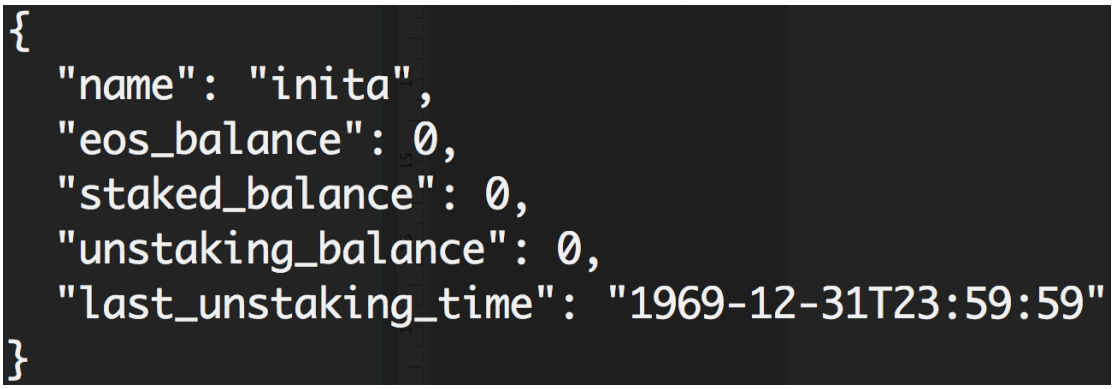
通过 eosc 调用 eos

此时已经可以通过 eosc 调用 eos。

./eosc info 可以查看目前的最新区块信息，例如：

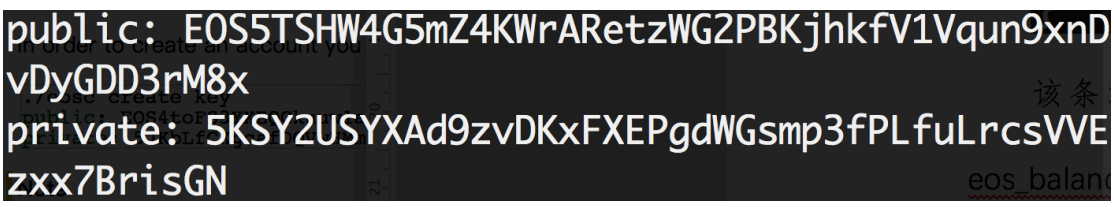


./eosc account inita 可以查看账号 inita 的信息。在配置文件 config.ini 和初始文件 genesis.json 中预制了 inita-initu 共 21 个账号作为 producer。



该条测试网络中内置了一个“eos”账号，拥有 9000000000000000 eos_balance。

用户可以通过./eosc create key 生成密钥对：



然后通过任意一个 producer 注册账号：

./eosc create account initb oraclechain
EOS5TSHW4G5mZ4KWrARetzWG2PBKjhkfV1Vqun9xnDvDyGDD3rM8x
EOS8TDzn2gkmMzBv56xLA8TFxVd5Sm4stuT8Nq3fFxCjhoz7GMYB5

两个公钥分别为 OWNERKEY 和 ACTIVEKEY。

随后可以./eosc account oraclechain 查看账号信息：


```
{
  "name": "oraclechain",
  "eos_balance": 0,
  "staked_balance": 1,
  "unstaking_balance": 0,
  "last_unstaking_time": "1969-12-31T23:59:59"
}
```


调用./eosc transfer eos oraclechain 1000 可以完成账号间的转账，然后再次查看账号：


```
{
  "name": "oraclechain",
  "eos_balance": 1000,
  "staked_balance": 1,
  "unstaking_balance": 0,
  "last_unstaking_time": "1969-12-31T23:59:59"
}
```


至此 EOS 的测试版本就已经跑起来了。开发者还可以通过 RPC 直接访问 eosd：

1. curl http://127.0.0.1:8888/v1/chain/get_info
2. curl http://localhost:8888/v1/chain/get_block -X POST -d '{"block_num_or_id":5}'
3. curl http://localhost:8888/v1/chain/get_block -X POST -d '{"block_num_or_id":0000000445a9f27898383fd7de32835d5d6a978cc14ce40d9f327b5329de796b}'
4. curl http://localhost:8888/v1/chain/push_transaction -X POST -d '{"refBlockNum":"5","refBlockPrefix":"27728114","expiration":"2017-07-

18T22:28:49","scope":["initb","initc"],"messages":[{"code":"currency","type":"transfer","recipients":["initb","initc"],"authorization":[{"account":"initb","permission":"active"}],"data":"c9252a0000000000050f14dc29000000d00700000000000008454f530000000000"}],"signatures":[],"authorizations":[]}'

目前 EOS 还没有提供签名机制，账户权限机制也还不够完善。相信 EOS 团队会在近期补充相关内容。

OracleChain 团队将在下一篇文章中对 EOS 的智能合约进行更详细的分析，相信能更进一步帮助开发者认识 EOS。

OracleChain 团队将第一时间持续更近 EOS 开发进度，陆续提供更多 EOS 功能的编译、测试用例。

开发者有任何技术问题可以在欧链科技社区里向 OracleChain 团队提问，或者发送邮件到 contact@oraclechain.io。

OracleChain 团队
2017 年 7 月 28 日