1. eos平台开发dapp的环境搭建

1.1. eos 编译与安装

安装eos对硬件和操作系统有一定要求，具体可以看到官网介绍

- Amazon 2017.09 and higher
- Centos 7
- Fedora 25 and higher (Fedora 27 recommended)
- Mint 18
- Ubuntu 16.04 (Ubuntu 16.10 recommended)
- Ubuntu 18.04
- MacOS Darwin 10.12 and higher (MacOS 10.13.x recommended)

而对于硬件则需要

- 7GB RAM free required
- 20GB Disk free required

具备以上条件以后，首先去github官网下载eos安装包

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

下载完成后进入eos目录，执行如下命令更新模块 `$ git submodule update --init --recursive`

然后执行脚本开始编译

```
$ ./eosio_build.sh
```

编译首先会更新对应操作系统的包管理器，以centos为例则会更新yum，检查是否存在有依赖的包

```
Checking YUM for installed dependencies.
Package git found.
Package autoconf found.
Package automake found.
Package bzip2 found.
Package libtool found.
Package ocaml.x86_64 found.
Package doxygen found.
Package graphviz-devel.x86_64 found.
Package libicu-devel.x86_64 found.
Package bzip2.x86_64 found.
Package bzip2-devel.x86_64 found.
Package openssl-devel.x86_64 found.
Package gmp-devel.x86_64 found.
Package python-devel.x86_64 found.
Package gettext-devel.x86_64 found.
No required YUM dependencies to install.
```

当满足所有依赖之后就会开始进行编译

```
ALL dependencies sucessfully found or installed . Installing EOSIO
```

在经过一阵长时间的编译操作之后，如果没有问题则会看到如下安装成功的画面

```
[100%] Built target nodeos

 _______  _______  _______ _________ _______
(  ____ \(  ___  )(  ____ \\__   __/(  ___  )
| (    \/| (   ) || (    \/   ) (   | (   ) |
| (__    | |   | || (_____    | |   | |   | |
|  __)   | |   | |(_____  )   | |   | |   | |
| (      | |   | |      ) |   | |   | |   | |
| (____/\| (___) |/\____) |___) (___| (___) |
(_______/(_______)\_______)\_______/(_______)

EOSIO has been successfully built. 00:01:09

To verify your installation run the following commands:
/root/opt/mongodb/bin/mongod -f /root/opt/mongodb/mongod.conf &
source /opt/rh/python33/enable
export PATH=${HOME}/opt/mongodb/bin:$PATH
cd /root/eos/build; make test

For more information:
EOSIO website: https://eos.io
EOSIO Telegram channel @ https://t.me/EOSProject
EOSIO resources: https://eos.io/resources/
EOSIO Stack Exchange: https://eosio.stackexchange.com
EOSIO wiki: https://github.com/EOSIO/eos/wiki
```

到此编译完成。然后运行eos自带的测试用例来验证安装结果。执行如下命令

```
$ /root/opt/mongodb/bin/mongod -f /root/opt/mongodb/mongod.conf &
$ source /opt/rh/python33/enable
$ export PATH=${HOME}/opt/mongodb/bin:$PATH
$ cd /root/eos/build; make test
```

等待测试结果，测试过程需要一定时间耐心等待。完成测试以后执行如下脚本

```
$ cd /root/eos/;./eosio_install.sh
```

执行成功后可看到如下画面

```
    Installing EOSIO Binary Symlinks

 _______  _______  _______ _________ _______
(  ____ \(  ___  )(  ____ \\__   __/(  ___  )
| (    \/| (   ) || (    \/   ) (   | (   ) |
| (__    | |   | || (_____    | |   | |   | |
|  __)   | |   | |(_____  )   | |   | |   | |
| (      | |   | |      ) |   | |   | |   | |
| (____/\| (___) |/\____) |___) (___| (___) |
(_______/(_______)\_______)\_______/(_______)
For more information:
EOSIO website: https://eos.io
EOSIO Telegram channel @ https://t.me/EOSProject
EOSIO resources: https://eos.io/resources/
EOSIO Stack Exchange: https://eosio.stackexchange.com
EOSIO wiki: https://github.com/EOSIO/eos/wiki
```

至此，编译完成。

1.2. 启动服务器

执行如下命令启动服务器

```
$ cd ~/eos/build/programs/keosd
$ keosd --http-server-address=localhost:8899
```

再打开一个新的命令行客户端

```
$ cd ~/eos/build/programs/nodeos
$ nodeos -e -p eosio --contracts-console --plugin eosio::chain_api_plugin --
plugin eosio::history_api_plugin
```

如果并非第一次执行上述命令，出错时需要添加参数 `--hard-replay`,该参数会花费一定时间同步之前生成的区块,再打开一个新的命令行客户端

```
$ alias cleos='~/eos/build/programs/cleos/cleos --wallet-
url=http://localhost:8899'
```

1.3. 创建钱包，密钥对，账户和代币

要在区块链中存储信息，我们需要一个用于标识数据和钱包的帐户来保护用于签署交易的密钥。请参阅此处了解EOSIO帐户和钱包概念概述

首先创建钱包

```
$ cd ~/eos;cleos wallet create -n mywallet --to-console
```

结果如下

```
Creating wallet: mywallet
Save password to use in the future to unlock this wallet.
Without password imported keys will not be retrievable.
"PW5HyLDXD6x97yuQwnk3Dci3ncsCveKYByAtgBVSbZfMXKhCmsh8c"
```

可选参数`--to-console`代表打印到终端，也可以添加参数`--file`保存到文件中,`-n`指定钱包名称。将打印到终端的密码记录下来作为备用。接下来生成密钥

```
$ cleos create key --to-console
```

记录显示在终端上的密钥对，再执行一次

```
$ cleos create key --to-console
```

然后将生成的密钥对中的私钥导入钱包

```
$ cleos wallet import -n mywallet --private-key {private key1}
$ cleos wallet import -n mywallet --private-key {private key2}
$ cleos wallet keys
```

查找config.ini文件，将配置项signature-provider的私钥导入钱包

```
$ find ~ -name config.ini $ cat ~/.local/share/eosio/nodes/config/config.ini
```

将私钥导入以后可以看到钱包中已经包含了3个公钥。使用刚才保存的两个公钥执行如下命令创建账号

```
$ cleos create account eosio myaccount {public key1} {public key2}
```

多创建几个账号

```
$ cleos create account eosio user {public key1} {public key2}
$ cleos create account eosio tester {public key1} {public key2}
$ cleos create account eosio eosio.token {public key1} {public key2}
```

在eosio.token账号上创建合约

```
$ cleos set contract eosio.token ~/eos/build/contracts/eosio.token -p
eosio.token
```

把合约推送到区块链

```
$ cleos push action eosio.token create '{"issuer":"eosio",
"maximum_supply":"1000000000.0000 SYS"}' -p eosio.token
```

做些单一操作测试，创建账号，发代币，转账

```
$ cleos create account eosio user1 ${public_key_1} ${public_key_2}
$ cleos push action eosio.token issue '[ "user", "100.0000 SYS", "memo" ]' -p
eosio
$ cleos push action eosio.token transfer '[ "user", "tester", "1.0000 SYS",
"m" ]' -p user
```

创建一个exchange账号，并建一个eosio.msig合约，和与代码在contracts/目录下，主要是允许多方异步签署单个交易

```
$ cleos set contract exchange ~/eos/build/contracts/eosio.msig -p exchange
```

备份钱包

```
$ mkdir backup-my-wallet/ $ cp -R ~/eosio-wallet ./backup-my-wallet/
```

1.4. 自己编写智能合约

在`contracts/`目录下提供了几个合约样本，直接使用`hello`合约的代码

```
#include <eosiolib/eosio.hpp>
#include <eosiolib/print.hpp>

using namespace eosio;

class hello : public contract {
  public:
      using contract::contract;

      [[eosio::action]]
      void hi( name user ) {
         print( "Hello, ", user);
      }
};

EOSIO_DISPATCH( hello, (hi))
```

进行如下测试

```
$ vim ~/eos/contracts/hello/hello.cpp
$ eosio-cpp -o hello.wasm hello.cpp --abigen\
```

执行此命令时发现无法找到`eosiocpp`命令，原来是需要安装`eosio.cdt`,执行如下命令安装

```
$ git clone --recursive https://github.com/eosio/eosio.cdt --branch v1.3.2 --
single-branch
$ cd eosio.cdt
$ ./build.sh
```

编译完成出现如下画面

```
     Scanning dependencies of target eosio
     [100%] Building CXX object \libraries/eosiolib/CMakeFiles/eosio.dir/eosiolib.cpp.o
     [100%] Linking CXX static library libeosio.a
     [100%] Built target eosio

      ___           ___           ___                       ___
     /  /\         /  /\         /  /\        ___          /  /\
    /  /:/_       /  /::\       /  /:/_      /  /\        /  /::\
   /  /:/ /\     /  /:/\:\     /  /:/ /\    /  /:/       /  /:/\:\
  /  /:/ /:/_   /  /:/  \:\   /  /:/ /::\  /__/::\      /  /:/  \:\
 /__/:/ /:/ /\ /__/:/ \__\:\ /__/:/ /:/\:\ \__\/\:\__  /__/:/ \__\:\
 \  \:\/:/ /:/ \  \:\ /  /:/ \  \:\/:/~/:/    \  \:\/\ \  \:\ /  /:/
  \  \::/ /:/   \  \:\  /:/   \  \::/ /:/      \__\::/  \  \:\  /:/
   \  \:\/:/     \  \:\/:/     \__\/ /:/       /__/:/    \  \:\/:/
    \  \::/       \  \::/        /__/:/        \__\/      \  \::/
     \__\/         \__\/         \__\/                     \__\/
For more information:
EOSIO website: https://eos.io
```

执行如下命令安装

```
$ ./install.sh
```

出现如下画面安装成功

```
    Installing EOSIO.CDT Binary Symlinks

      ___           ___           ___                       ___
     /  /\         /  /\         /  /\        ___          /  /\
    /  /:/_       /  /::\       /  /:/_      /  /\        /  /::\
   /  /:/ /\     /  /:/\:\     /  /:/ /\    /  /:/       /  /:/\:\
  /  /:/ /:/_   /  /:/  \:\   /  /:/ /::\  /__/::\      /  /:/  \:\
 /__/:/ /:/ /\ /__/:/ \__\:\ /__/:/ /:/\:\ \__\/\:\__  /__/:/ \__\:\
 \  \:\/:/ /:/ \  \:\ /  /:/ \  \:\/:/~/:/    \  \:\/\ \  \:\ /  /:/
  \  \::/ /:/   \  \:\  /:/   \  \::/ /:/      \__\::/  \  \:\  /:/
   \  \:\/:/     \  \:\/:/     \__\/ /:/       /__/:/    \  \:\/:/
    \  \::/       \  \::/        /__/:/        \__\/      \  \::/
     \__\/         \__\/         \__\/                     \__\/
For more information:
EOSIO website: https://eos.io
```

创建账号`hello`(参考之前的命令),创建合约

```
$ cleos set contract hello ../hello -p hello@active
```

推送合约

```
$ cleos push action hello hi '["bob"]' -p user@active
```

可以看到如下结果

```
[root@localhost hello]# cleos push action hello hi '["bob"]' -p user@active
executed transaction:
a4f205dfa4b6e66837e8485baf8760a9a2c568dd8f55bdbda1437e92ecfffff2 104 bytes
500 us
# hello <= hello::hi {"user":"bob"}
>> hello, bob
```

这表明合约执行成功啦，然后我们修改`hello.cpp`文件，在方法体第一行加入`require_auth(user)`,代码如下

```
#include <eosiolib/eosio.hpp>
#include <eosiolib/print.hpp>

using namespace eosio;

class hello : public contract {
  public:
      using contract::contract;

      [[eosio::action]]
      void hi( name user ) {
         require_auth(user);
         print( "Hello, ", user);
      }
};

EOSIO_DISPATCH( hello, (hi))
```

重新编译生成可执行文件后重新推送会得到结果

```
[root@localhost hello]# cleos push action hello hi '["bob"]' -p tester@active
Error 3090004: Missing required authority
```

此时需要账号和用户名一致，修改推送命令

```
$ cleos push action hello hi '["tester"]' -p tester@active
```

可得到与刚才类似的结果

```
[root@localhost hello]# cleos push action hello hi '["tester"]' -p tester@active
executed transaction:
72cdff031464492bd06a4d3337ef2b1a33588f638bc482a3aaaa539e554a4fac 104 bytes
540 us
# hello <= hello::hi {"user":"tester"}
>> hello, tester
```

然后执行如下命令关闭进行

```
$ pkill keosd && pkill nodeos
```

简介到此结束