18342025_胡鹏飞_热身报告

阶段要求

- 使用已有的开源区块链系统FISCO-BCOS，完成私有链的搭建以及新节点的加入。（截图说明搭建流程）
- 自行编写一个智能合约并部署到私有链上，同时完成合约调用。（截图说明部署流程）
- 使用命令查看一个区块，并对各个字段进行解释。

实验环境

操作系统：macOS Big Sur 11.0.1

实验过程

本次实验过程都是按照 `FISCO BCOS` 官方文档的步骤进行环境搭建及实验的进行，官方文档链接：传送门

私有链的搭建

- 开发部署的工具 `build_chain.sh` 依赖 `openssl, curl` ，根据以下命令安装依赖：

  ```
  brew install openssl curl
  ```

```
[hupf@hupengfeideMacBook-Pro ~ % brew install openssl curl
Updating Homebrew...
^CWarning: You are using macOS 11.0.
We do not provide support for this pre-release version.
You will encounter build failures with some formulae.
Please create pull requests instead of asking for help on Homebrew's GitHub,
Discourse, Twitter or IRC. You are responsible for resolving any issues you
experience while you are running this pre-release version.

==> Downloading https://homebrew.bintray.com/bottles/openssl%401.1-1.1.1h.catali
==> Downloading from https://d29vzk4ow07wi7.cloudfront.net/4e5357c0cfd55cfa4ef0b
######################################################################## 100.0%
==> Pouring openssl@1.1-1.1.1h.catalina.bottle.tar.gz
==> Caveats
A CA file has been bootstrapped using certificates from the system
keychain. To add additional certificates, place .pem files in
  /usr/local/etc/openssl@1.1/certs

and run
  /usr/local/opt/openssl@1.1/bin/c_rehash

openssl@1.1 is keg-only, which means it was not symlinked into /usr/local,
because macOS provides LibreSSL.

If you need to have openssl@1.1 first in your PATH run:
  echo 'export PATH="/usr/local/opt/openssl@1.1/bin:$PATH"' >> ~/.zshrc

For compilers to find openssl@1.1 you may need to set:
  export LDFLAGS="-L/usr/local/opt/openssl@1.1/lib"
  export CPPFLAGS="-I/usr/local/opt/openssl@1.1/include"

==> Summary
🍺  /usr/local/Cellar/openssl@1.1/1.1.1h: 8,067 files, 18.5MB
==> Downloading https://homebrew.bintray.com/bottles/curl-7.73.0.catalina.bottle
==> Downloading from https://d29vzk4ow07wi7.cloudfront.net/98f3bd49f4eae8638edc3
######################################################################## 100.0%
==> Pouring curl-7.73.0.catalina.bottle.tar.gz
==> Caveats
curl is keg-only, which means it was not symlinked into /usr/local,
because macOS already provides this software and installing another version in
parallel can cause all kinds of trouble.

If you need to have curl first in your PATH run:
  echo 'export PATH="/usr/local/opt/curl/bin:$PATH"' >> ~/.zshrc

For compilers to find curl you may need to set:
  export LDFLAGS="-L/usr/local/opt/curl/lib"
  export CPPFLAGS="-I/usr/local/opt/curl/include"
```

由上图可知成功安装好了依赖环境

- 然后需要创建 `fisco` 操作目录，并且下载安装脚本

```
## 创建操作目录
cd ~ && mkdir -p fisco && cd fisco

## 下载脚本
curl -#LO https://github.com/FISCO-BCOS/FISCO-
BCOS/releases/download/v2.7.0/build_chain.sh && chmod u+x build_chain.sh
```

```
[hupf@hupengfeideMacBook-Pro fisco % curl -#LO https://gitee.com/FISCO-BCOS/FISCO]
-BCOS/raw/master/tools/build_chain.sh && chmod u+x build_chain.sh
  -=#=-#   #     #
[hupf@hupengfeideMacBook-Pro fisco % ls                                          ]
build_chain.sh
```

- 安装好脚本后，开始搭建单群组四节点的联盟链，还需要确保 `30300~30303，20200~20203，8545~8548` 端口没有被占用。在 mac 中用以下的命令即可进行检测端口的使用情况：

  `lsof -i:端口号`

```
[hupf@hupengfeideMacBook-Pro fisco % lsof -i:8545
[hupf@hupengfeideMacBook-Pro fisco % lsof -i:8546
[hupf@hupengfeideMacBook-Pro fisco % lsof -i:8547
[hupf@hupengfeideMacBook-Pro fisco % lsof -i:8548
[hupf@hupengfeideMacBook-Pro fisco % lsof -i:30300
[hupf@hupengfeideMacBook-Pro fisco % lsof -i:30301
[hupf@hupengfeideMacBook-Pro fisco % lsof -i:30302
[hupf@hupengfeideMacBook-Pro fisco % lsof -i:30303
[hupf@hupengfeideMacBook-Pro fisco % lsof -i:20200
[hupf@hupengfeideMacBook-Pro fisco % lsof -i:20201
[hupf@hupengfeideMacBook-Pro fisco % lsof -i:20202
[hupf@hupengfeideMacBook-Pro fisco % lsof -i:20203
```

由上图可知该端口并没有被占用，所以可以开始配置：

`bash build_chain.sh -l 127.0.0.1:4 -p 30300,20200,8545`

```
hupf@hupengfeideMacBook-Pro fisco % bash build_chain.sh -l 127.0.0.1:4 -p 30300,20200,8545
[INFO] Downloading fisco-bcos binary from https://github.com/FISCO-BCOS/FISCO-BCOS/releases/download/v2.7.0/fisco-bcos-macOS.tar.gz ...
############################################################################## 100.0%############################################################################################## 100.0%
================================================================
Generating CA key...
================================================================
Generating keys and certificates ...
Processing IP=127.0.0.1 Total=4 Agency=agency Groups=1
================================================================
Generating configuration files ...
Processing IP=127.0.0.1 Total=4 Agency=agency Groups=1
================================================================
[INFO] Start Port      : 30300 20200 8545
[INFO] Server IP       : 127.0.0.1:4
[INFO] Output Dir      : /Users/hupf/fisco/nodes
[INFO] CA Path         : /Users/hupf/fisco/nodes/cert/
================================================================
[INFO] Execute the download_console.sh script in directory named by IP to get FISCO-BCOS console.
e.g.  bash /Users/hupf/fisco/nodes/127.0.0.1/download_console.sh -f
================================================================
[INFO] All completed. Files in /Users/hupf/fisco/nodes
hupf@hupengfeideMacBook-Pro fisco %
```

命令执行成功会输出 `All completed`。

- 接下来就启动 `FISCO BCOS` 链：

  `bash nodes/127.0.0.1/start_all.sh`

  成功启动会有下面的提示信息：

  ```
  hupf@hupengfeideMacBook-Pro fisco % bash nodes/127.0.0.1/start_all.sh
  try to start node0
  try to start node1
  try to start node2
  try to start node3
   node2 start successfully
   node0 start successfully
   node1 start successfully
   node3 start successfully
  ```

- 启动后开始检查进程：

  `ps -ef | grep -v grep | grep fisco-bcos`

  可以发现有四个进程数目，说明启动成功

```
hupf@hupengfeideMacBook-Pro fisco % ps -ef | grep -v grep | grep fisco-bcos
  501  3451     1   0  9:47上午 ttys000    0:04.16 /Users/hupf/fisco/nodes/127.0.0.1/node2/../fisco-bcos -c config.ini
  501  3453     1   0  9:47上午 ttys000    0:04.13 /Users/hupf/fisco/nodes/127.0.0.1/node1/../fisco-bcos -c config.ini
  501  3454     1   0  9:47上午 ttys000    0:04.18 /Users/hupf/fisco/nodes/127.0.0.1/node0/../fisco-bcos -c config.ini
  501  3457     1   0  9:47上午 ttys000    0:04.14 /Users/hupf/fisco/nodes/127.0.0.1/node3/../fisco-bcos -c config.ini
```

配置使用控制台

- 首先需要配置 `java` 环境，即在 java 官网 中可以下载

```
[hupf@hupengfeideMacBook-Pro ~ % java -version
java version "1.8.0_261"
Java(TM) SE Runtime Environment (build 1.8.0_261-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.261-b12, mixed mode)
hupf@hupengfeideMacBook-Pro ~ %
```

配置好环境后需要获取控制台，命令如下：

`cd ~/fisco && curl -#LO https://github.com/FISCO-BCOS/console/releases/download/v2.7.0/download_console.sh && bash download_console.sh`

```
[hupf@hupengfeideMacBook-Pro fisco % cd ~/fisco && curl -#LO https://gitee.com/FISCO-BCOS]
/console/raw/master/tools/download_console.sh
-#O#-#   #
[hupf@hupengfeideMacBook-Pro fisco % ls                                                 ]
build_chain.sh          download_console.sh     nodes
```

成功获取控制台后，需要配置控制台的文件，和控制台证书：

```
# 最新版本控制台使用如下命令拷贝配置文件
cp -n console/conf/config-example.toml console/conf/config.toml

cp -r nodes/127.0.0.1/sdk/* console/conf/
```

```
[hupf@hupengfeideMacBook-Pro fisco % cp -n console/conf/config-example.toml console/conf/config.toml
hupf@hupengfeideMacBook-Pro fisco % cp -r nodes/127.0.0.1/sdk/* console/conf/
```

- 接下来可以启动并使用控制台：

  `cd ~/fisco/console && bash start.sh`

  这里我在第一遍尝试的时候发现并不可以成功启动：

```
[hupf@hupengfeideMacBook-Pro console % cd ~/fisco/console && bash start.sh        ]
create BcosSDK failed, error info: init channel network error:  Failed to connect
 to all the nodes! errorMessage:
 ssl handshake failed:/127.0.0.1:20200! Please check the certificate and ensure t
hat the SDK and the node are in the same agency!
 ssl handshake failed:/127.0.0.1:20201! Please check the certificate and ensure t
hat the SDK and the node are in the same agency!
```

  后来发现是 `java` 版本的问题，后来重新装了 `java` 可以成功运行：

```
[hupf@hupengfeideMacBook-Pro console % bash start.sh
================================================================================================
Welcome to FISCO BCOS console(2.7.0)!
Type 'help' or 'h' for help. Type 'quit' or 'q' to quit console.
 ________ ______  ______   ______   ______       _______  ______   ______   ______
|        |      \/      \ /      \ /      \     |       \ /      \ /      \ /      \
| $$$$$$$$\$$$$$|  $$$$$$|  $$$$$$|  $$$$$$\    | $$$$$$$|  $$$$$$|  $$$$$$|  $$$$$$\
| $$__     | $$ | $$___\$| $$   \$| $$  | $$    | $$__/ $| $$   \$| $$  | $| $$___\$$
| $$  \    | $$  \$$    \| $$      | $$  | $$    | $$    $| $$      | $$  | $$\$$    \
| $$$$$    | $$  _\$$$$$$| $$   __| $$  | $$    | $$$$$$$| $$   __| $$  | $$_\$$$$$$\
| $$      _| $$_|  \__| $| $$__/  | $$__/ $$    | $$__/ $| $$__/  | $$__/ $|  \__| $$
| $$     |   $$ \\$$    $$\$$    $$\$$    $$    | $$    $$\$$    $$\$$    $$\$$    $$
 \$$      \$$$$$$ \$$$$$$  \$$$$$$  \$$$$$$      \$$$$$$$  \$$$$$$  \$$$$$$  \$$$$$$

================================================================================================
[group:1]>
```

  不输成功后可以简单进行命令的测试，看看是否能够成功使用：

```
[[group:1]> getNodeVersion
ClientVersion{
    version='2.7.0',
    supportedVersion='2.7.0',
    chainId='1',
    buildTime='20201126 08:04:59',
    buildType='Darwin/appleclang/RelWithDebInfo',
    gitBranch='HEAD',
    gitCommitHash='0bc47666979df4766723adf1ab9c5d80f5e40537'
}

[[group:1]> getPeers
[
    PeerInfo{
        nodeID='a641ec9c514957e2bd6ac4867d16fc0ea6590eacf2425d86d012a0330493ed04720f9ded466382a94
beaea1a78fb6a9ffae64c8f9d78fab',
        iPAndPort='127.0.0.1:51141',
        node='node2',
        agency='agency',
        topic='[

        ]'
    },
    PeerInfo{
        nodeID='9ac15626a1d71a10ab1803c6e22e6a573ababeb32a69f4f8eb4581f204866d922f98a9cc0222abc24
2e7019972c22bc40a828a35cf8d8b4',
        iPAndPort='127.0.0.1:51171',
        node='node0',
        agency='agency',
        topic='[
            _block_notify_1
        ]'
    }
]

[[group:1]> quit
```

新节点的加入

- 首先获取证书生成新的脚本：

  `curl -#LO https://raw.githubusercontent.com/FISCO-BCOS/FISCO-BCOS/master/tools/gen_node_cert.sh`

```
[hupf@hupengfeideMacBook-Pro 127.0.0.1 % curl -#LO https://gitee.com/FISCO-BCOS/FISCO-BCOS/raw/master/tools/gen_nod
e_cert.sh
#=#=- #   #
hupf@hupengfeideMacBook-Pro 127.0.0.1 %
```

  然后需要生成新节点私钥证书：

  `bash gen_node_cert.sh -c ../cert/agency -o newNode`

```
hupf@hupengfeideMacBook-Pro 127.0.0.1 % bash gen_node_cert.sh -c ../cert/agency -o newNode

================================================================
[INFO] Cert Path    : ../cert/agency
[INFO] Output Dir   : newNode
================================================================
[INFO] All completed. Files in newNode
```

- 然后需要准备配置文件，需要拷贝 `node0` 中的配置文件与脚本工具到新的节点中

```
cp node0/config.ini newNode/config.ini
cp node0/conf/group.1.genesis newNode/conf/group.1.genesis
cp node0/conf/group.1.ini newNode/conf/group.1.ini
cp node0/*.sh newNode/
cp -r node0/scripts newNode/
```

更新 `newNode/config.ini` 中监听的 IP 和端口，对于 `[rpc]` 模块，修改 `listen_ip`、`channel_listen_port` 和 `jsonrpc_listen_port`；对于 `[p2p]` 模块，修改 `listen_port`

```
[rpc]
    channel_listen_ip=0.0.0.0
    channel_listen_port=20204
    jsonrpc_listen_ip=127.0.0.1
    jsonrpc_listen_port=8549
[p2p]
    listen_ip=0.0.0.0
    listen_port=30304
    ; nodes to connect
    node.0=127.0.0.1:30300
    node.1=127.0.0.1:30301
    node.2=127.0.0.1:30302
    node.3=127.0.0.1:30303
```

将新节点的 P2P 配置中的 IP 和 Port 加入原有节点的 `config.ini` 中的 [p2p] 字段。

然后启动新的节点，观察情况：

`newNode/start.sh`

```
hupf@hupengfeideMacBook-Pro 127.0.0.1 % cd newNode
hupf@hupengfeideMacBook-Pro newNode % bash start.sh
 newNode start successfully
hupf@hupengfeideMacBook-Pro newNode %
```

然后进行查看新节点的 `NodeID`：

`cat newNode/conf/node.nodeid`

```
hupf@hupengfeideMacBook-Pro newNode % cd ..
hupf@hupengfeideMacBook-Pro 127.0.0.1 % cat newNode/conf/node.nodeid
42d0ad671937ae1f50401948146970b17631f5c788cea6323d7a79451f6c05fa6f66bbdfc413908b032641df601306697eaf8cba1c0b20609d
52f6acc321ed48
```

通过控制台发现新的节点在列表中:

```
[group:1]> getNodeIDList
[
    ae66b2baab1d4ce23efed0247547ee4cb1512c80f30a00a7991b0fd69ba2ec2f9cfb24cdda9ed4c7935cd68cb574305410d30f7b154c04
bfe133ae8dcb097221,
    a641ec9c514957e2bd6ac4867d16fc0ea6590eacf2425d86d012a0330493ed04720f9ded466382a9421f01612f9650313abeaea1a78fb6
a9ffae64c8f9d78fab,
    7525df7a5b8fa2fcf66f3951d6d335ca7f193a73c3fbcefdc7ea71149302ac8dbc549294d0a93ab01c9f0b38f569a4e278a8df5d426754
ceb8699e489635293d,
    9ac15626a1d71a10ab1803c6e22e6a573ababeb32a69f4f8eb4581f204866d922f98a9cc0222abc24a04e79eef96ba362b2e7019972c22
bc40a828a35cf8d8b4,
    42d0ad671937ae1f50401948146970b17631f5c788cea6323d7a79451f6c05fa6f66bbdfc413908b032641df601306697eaf8cba1c0b20
609d52f6acc321ed48
]
```

然后我们将新的节点加入到公式节点中

```
[group:1]> addSealer 42d0ad671937ae1f50401948146970b17631f5c788cea6323d7a79451f6c05fa6f66bbdfc413908b032641df60130
6697eaf8cba1c0b20609d52f6acc321ed48
{
    "code":1,
    "msg":"Success"
}

[group:1]>
```

发现插入成功，然后观察公式节点，发现确实如此：

```
[group:1]> getSealerList
[
    7525df7a5b8fa2fcf66f3951d6d335ca7f193a73c3fbcefdc7ea71149302ac8dbc549294d0a93ab01c9f0b38f569a4e278a8df5d426754ceb8699e489635293d,
    9ac15626a1d71a10ab1803c6e22e6a573ababeb32a69f4f8eb4581f204866d922f98a9cc0222abc24a04e79eef96ba362b2e7019972c22bc40a828a35cf8d8b4,
    a641ec9c514957e2bd6ac4867d16fc0ea6590eacf2425d86d012a0330493ed04720f9ded466382a9421f01612f9650313abeaea1a78fb6a9ffae64c8f9d78fab,
    ae66b2baab1d4ce23efed0247547ee4cb1512c80f30a00a7991b0fd69ba2ec2f9cfb24cdda9ed4c7935cd68cb574305410d30f7b154c04bfe133ae8dcb097221,
    42d0ad671937ae1f50401948146970b17631f5c788cea6323d7a79451f6c05fa6f66bbdfc413908b032641df601306697eaf8cba1c0b20609d52f6acc321ed48
]
```

至此成功插入新的节点

编写智能合约

- 首先编写一个简单的 `HelloWorld` 合约，该合约提供两个接口，分别是 `get()` 和 `set()`，用于获取/设置合约变量 `name`。合约内容如下:

```
pragma solidity ^0.4.24;

contract HelloWorld {
    string name;

    function HelloWorld() {
        name = "Hello, World!";
    }

    function get()constant returns(string) {
        return name;
    }

    function set(string n) {
        name = n;
    }
}
```

- 成功写好了合约后，可以进行命令的部署，部署成功后即刻返回合约的地址：

```
[group:1]> deploy HelloWorld
transaction hash: 0x31af016715ab5bd8ab15eb6a4ebdf7e7f99baa231e1a9a30b48795bd6b953c61
contract address: 0x8950d0bdb1720c5ddf668aeafcb9d35c0029f641
```

可以看到成功部署了该合约

- 然后开始调用该合约，应 `call` 命令来调取两个接口 `get` 和 `set`
  - `get` 接口：

```
[group:1]> call HelloWorld 0x8950d0bdb1720c5ddf668aeafcb9d35c0029f641 get
-------------------------------------------------------------------------------------------
Return code: 0
description: transaction executed successfully
Return message: Success
-------------------------------------------------------------------------------------------
Return values:
[
    "Hello,World!"
]
-------------------------------------------------------------------------------------------
```

- `set 接口`：

```
[group:1]> call HelloWorld 0x8950d0bdb1720c5ddf668aeafcb9d35c0029f641 set "Hello SYSU"
transaction hash: 0x4ec3bae10287174456caa0717cd76eced81c85e4cf7db1eb1e385ad1be101ee4
-------------------------------------------------------------------------------------------
transaction status: 0x0
description: transaction executed successfully
-------------------------------------------------------------------------------------------
Output
Receipt message: Success
Return message: Success
Return value: []
-------------------------------------------------------------------------------------------
Event logs
Event: {}
```

成功调用合约中的两个接口后，变量是否改变：

```
[group:1]> call HelloWorld 0x8950d0bdb1720c5ddf668aeafcb9d35c0029f641 get
-------------------------------------------------------------------------------------------
Return code: 0
description: transaction executed successfully
Return message: Success
-------------------------------------------------------------------------------------------
Return values:
[
    "Hello SYSU"
]
-------------------------------------------------------------------------------------------
```

可以发现成功部署智能合约，并且实现了合约的调用。

命令查看区块

- `getBlockNumber`：查看区块的高度
- `getBlockHeaderByHash`根据区块哈希查询区块头信息
  - `dbHash`：主要是用来比较MongoDB数据是否一致
  - `extraData`：字符串。当前块的extra data字段。
  - `gasLimit`：Number，当前区块允许使用的最大gas。
  - `gasUsed`：当前区块累计使用的总的gas。
  - `hash`：字符串，区块的哈希串。当这个区块处于 pending 将会返回 null。
  - `logsBloom`：字符串，区块日志的布隆过滤器。当这个区块处于 pending 将会返回 null。
  - `number`：区块号。当这个区块处于 pending 将会返回 null。
  - `parentHash`：字符串，32字节的父区块的哈希值
  - `receiptsRoot`：存储的是交易回执内容
  - `sealer`：共识节点
  - `sealerList`：共识节点列表
  - `stateRoot`：字符串，32字节。区块的最终状态前缀树的根。
  - `timestamp`：Number，区块打包时的 unix 时间戳。
  - `transactionsRoot`：字符串，32字节，区块的交易前缀树的根。
- `getBlockHeaderByNumber`：根据区块高度查询区块头信息(与上面重复的不再解释)
  - `transactions`：数组。交易对象。或者是32字节的交易哈希。
- `getBlockByHash`：根据区块哈希查询区块信息
- `getBlockByNumber`：根据区块高度查询区块信息
- `getBlockHashByNumber`：通过区块高度获取区块的哈希值

实验总结

本次实验是对环境的基础配置，以及对于控制台各种命令的掌握。以及一些操作如私有链的搭建过程，新节点的加入过程，部署智能合约并且完成合约调用。对于区块的各个字段也有初步的了解。由于老师说 mac 端在后续的实验中可能会有 bug，所以我在 Centos 和 Ubuntu 上都配置好了环境，以便后续的进展。下一步计划是完成各个功能的测试。