School of Data and Computer Science, Sun Yat-sen University

区块链大作业：热身报告

18308045 Zhengyang Gu

2020 年 11 月 27 日

目录

1 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）

1.1 单群组FISCO BCOS联盟链的搭建

1.1.1 准备环境

1. 安装依赖

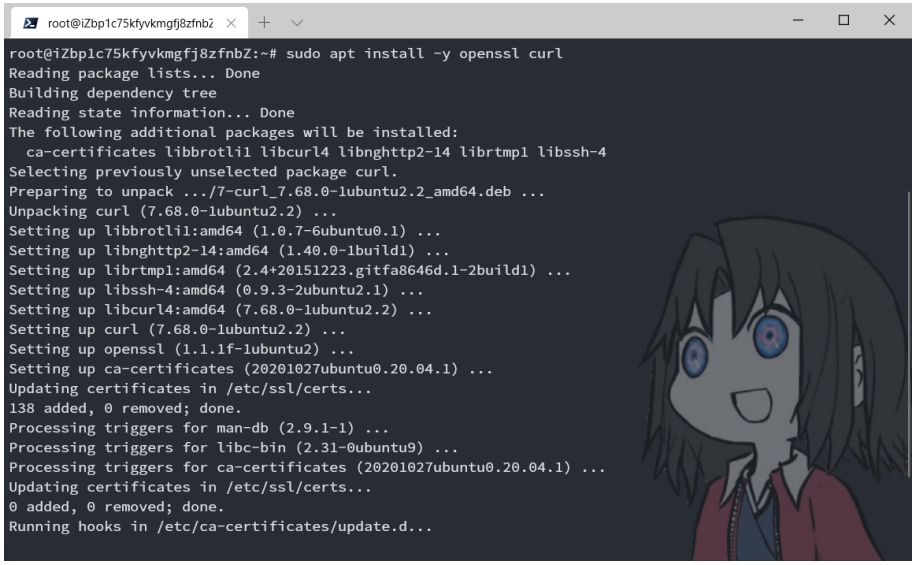

图 1: 安装ubuntu依赖

2. 创建操作目录

图 2: 创建操作目录

3. 下载build_chain.sh脚本

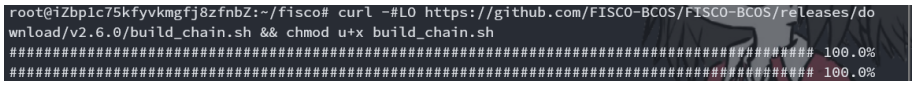

图 3: 下载build_chain.sh脚本

1.1.2 搭建单群组4节点联盟链

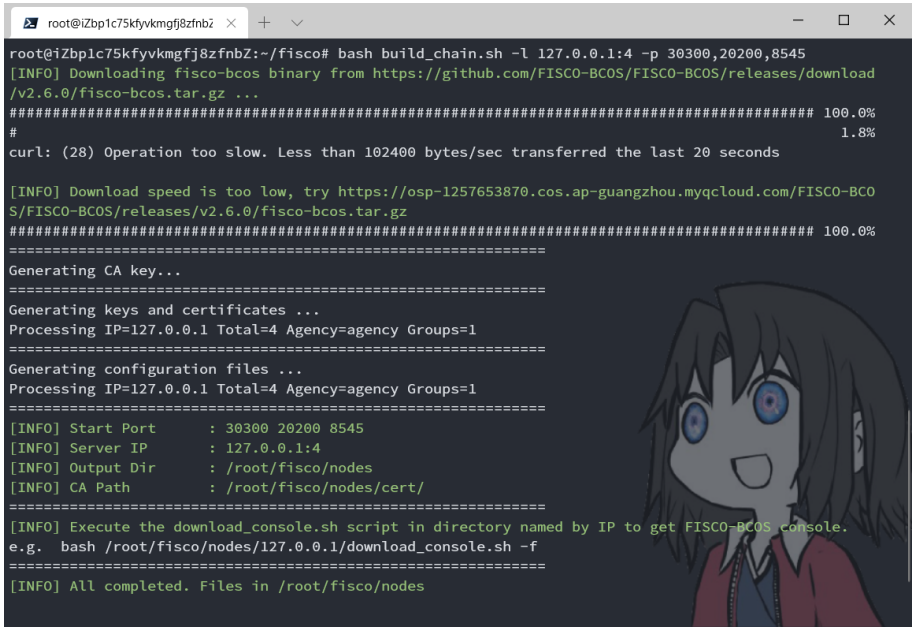

图 4: 搭建单群组4节点联盟链

1.1.3 启动FISCO BCOS链

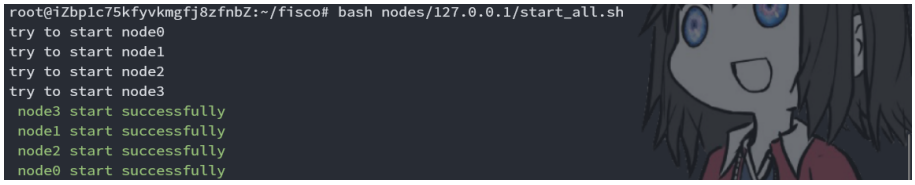

图 5: 启动FISCO BCOS链

1.1.4 检查进程

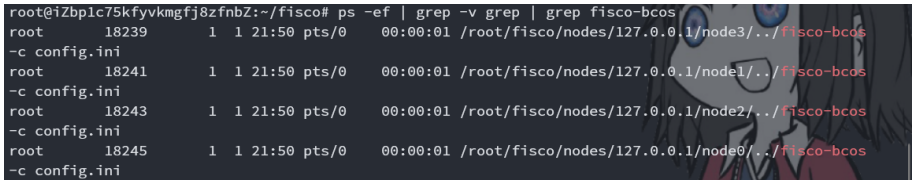

图 6: 检查进程

1.1.5 检查日志输出

1. 查看节点node0链接的节点数

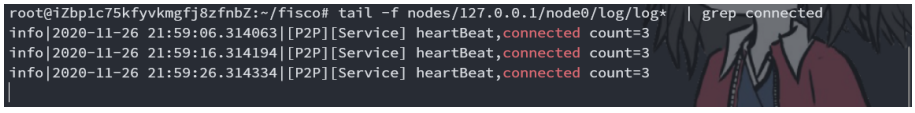

图 7: 查看节点node0链接的节点数

2. 检查是否在共识

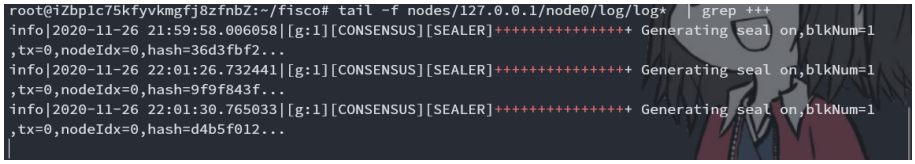

图 8: 检查是否在共识

1.2 配置及使用控制台

1.2.1 准备依赖

1. 安装java

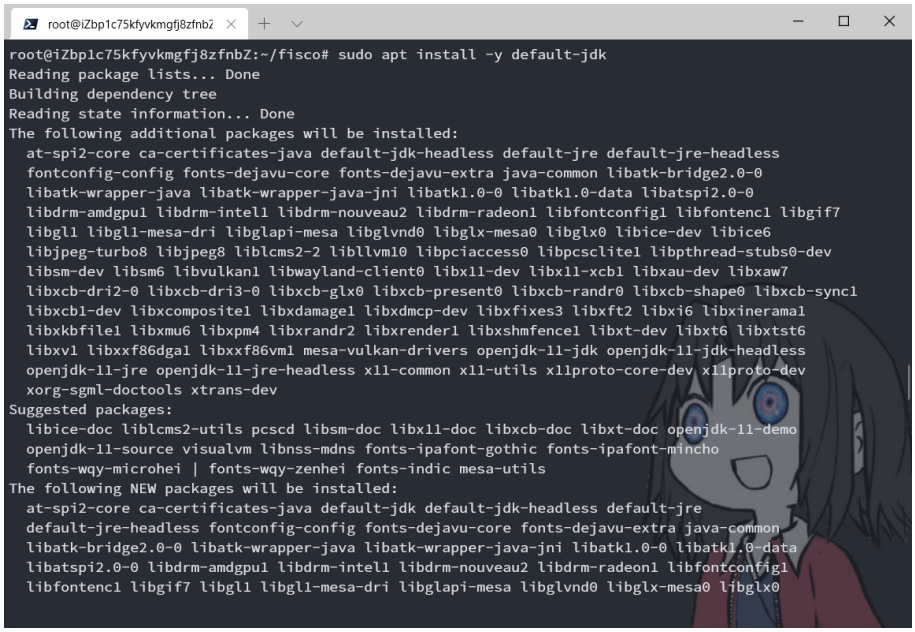

图 9: 安装java

2. 获取控制台并回到fisco目录

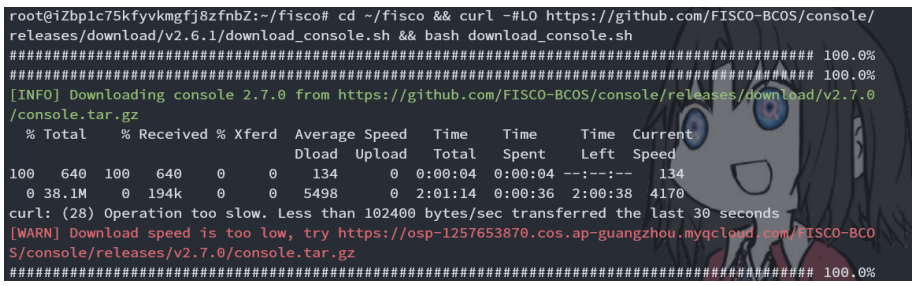

图 10: 获取控制台并回到fisco目录

3. 拷贝控制台配置文件

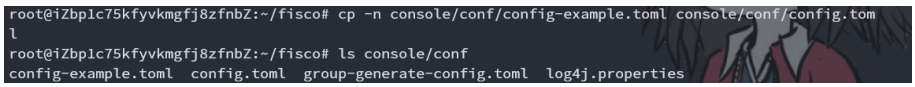

图 11: 拷贝控制台配置文件

4. 配置控制台证书

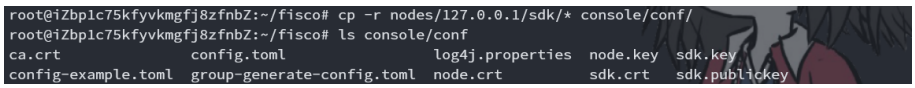

图 12: 配置控制台证书

1.2.2 启动控制台

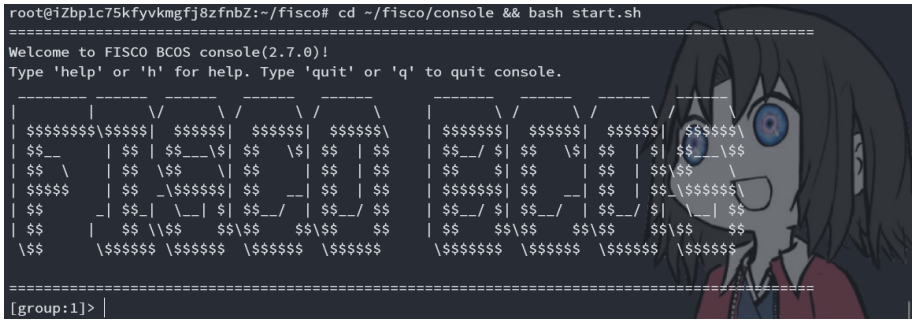

图 13: 启动控制台

1.2.3 使用控制台获取信息

1. 获取客户端版本

图 14: 获取客户端版本

2. 获取节点链接信息

图 15: 获取节点链接信息

1.3 群组新增节点

1.3.1 为新节点生成私钥证书

1. 获取证书生成脚本

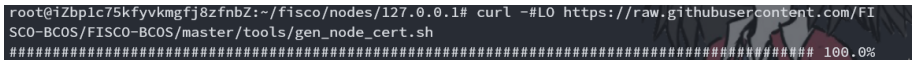

图 16: 获取证书生成脚本

2. 生成新节点私钥证书

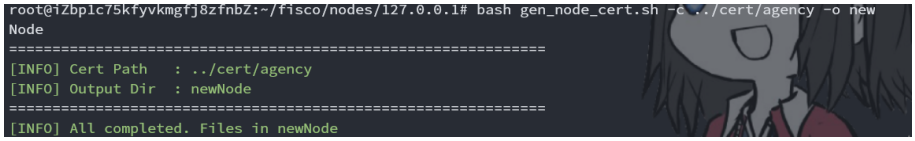

图 17: 生成新节点私钥证书

1.3.2 准备配置文件

1. 拷贝群组1中节点node0配置文件与工具脚本

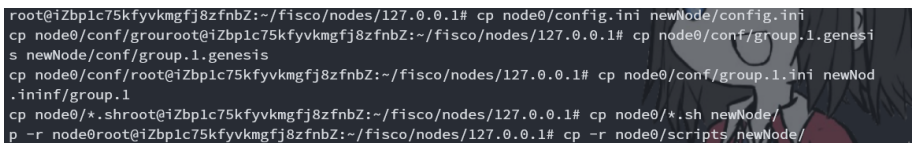

图 18: 拷贝群组1中节点node0配置文件与工具脚本

2. 更新newNode/config.ini中监听的IP和端口

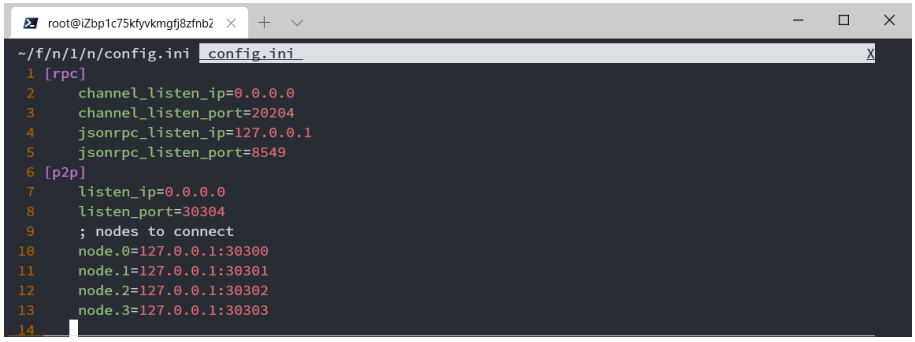

图 19: 更新newNode/config.ini中监听的IP和端口

3. 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

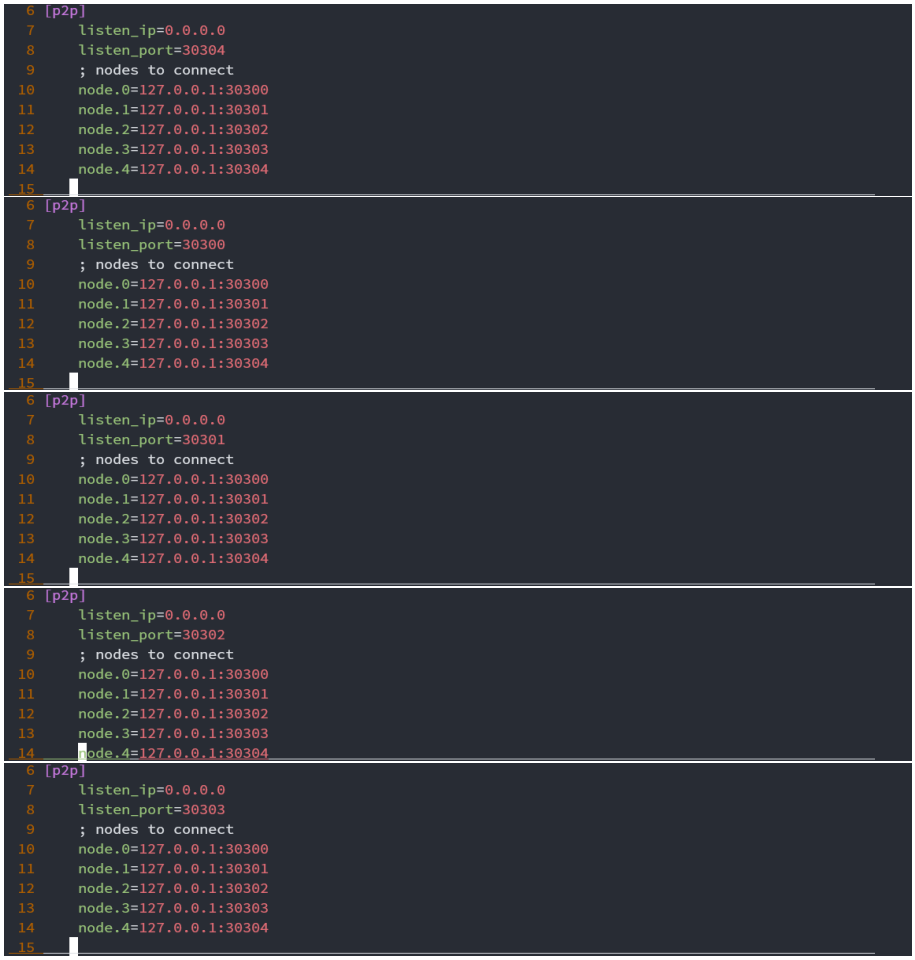

图 20: 将新节点的P2P配置中的IP和Port加入原有节点的config.ini中的[p2p]字段

1.3.3 启动新节点并加入群组

1. 启动新节点，执行newNode/start.sh

图 21: 启动新节点，执行newNode/start.sh

2. 通过console将新节点加入群组1

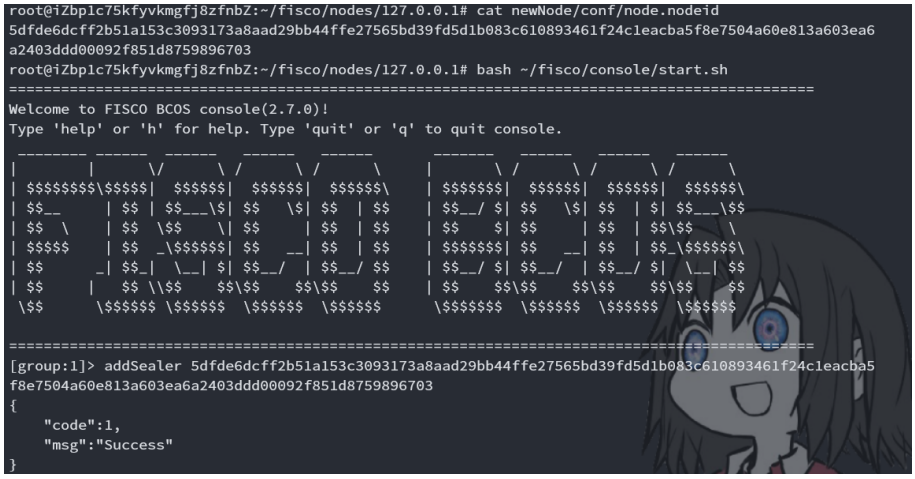

图 22: 通过console将新节点加入群组1

3. 检查连接和共识

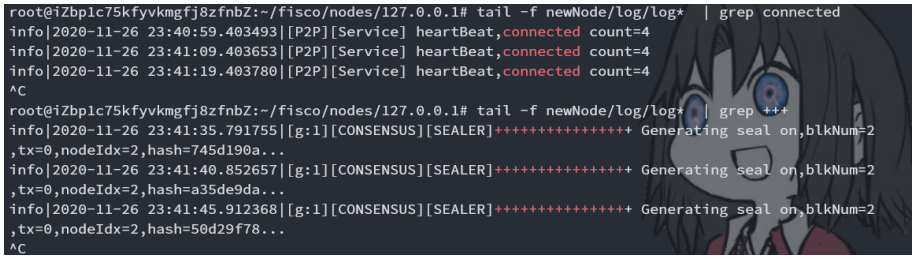

图 23: 检查连接和共识

2 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）

2.1 Upvote合约

2.1.1 提供接口

1. **publish(content_names)**：参数是内容名字的数组，发布内容，只有合约中唯一的发布者可以调用。

2. **giveVotesToUser(user)**：参数是用户地址，给该用户10个点赞机会，只有合约中唯一的发布者可以调用，每个用户只能被授予1次。

3. **upvote(name, votes)**：参数是内容名和点赞数，给某个内容点赞若干次。

4. **winner()**：返回当前票数最高的内容名。

2.1.2 合约内容

```
pragma solidity >=0.4.24;
```

```
2
3  contract Upvote
4  {
5      struct User
6      {
7          uint votes;
8          bool given;
9      }
10
11     struct Content
12     {
13         string name;
14         uint votes;
15     }
16
17     address public publisher;
18     mapping(address => User) public users;
19     Content[] public contents;
20
21     constructor() public
22     {
23         publisher = msg.sender;
24     }
25
26     function publish(string memory name) public
27     {
28         require(msg.sender == publisher, "Only the publisher can
       publish contents.");
29         for (uint i = 0; i < contents.length; i++)
30         {
31             require(keccak256(name) != keccak256(contents[i].
       name), "The content already exists.");
32         }
33         contents.push(Content({name: name, votes: 0}));
34     }
35
36     function giveVotesToUser(address user) public
37     {
38         require(msg.sender == publisher, "Only the publisher can
       give votes to a user.");
```

```
39          require(users[user].given == false, "The user has been
    given votes.");
40          users[user].votes = 10;
41      }
42
43      function upvote(string memory name, uint votes) public
44      {
45          User storage sender = users[msg.sender];
46          require(sender.votes >= votes, "Don't have enough votes.
    ");
47          uint i = 0;
48          for (; i < contents.length; i++)
49          {
50              if (keccak256(name) == keccak256(contents[i].name))
51              {
52                  sender.votes -= votes;
53                  contents[i].votes += votes;
54                  break;
55              }
56          }
57          require(i < contents.length, "The content doesn't exist.
    ");
58      }
59
60      function winner() public view returns(string memory)
61      {
62          uint max_votes = 0;
63          string storage name;
64          for (uint i = 0; i < contents.length; i++)
65          {
66              if (max_votes < contents[i].votes)
67              {
68                  max_votes = contents[i].votes;
69                  name = contents[i].name;
70              }
71          }
72          return name;
73      }
74  }
```

2.2 部署Upvote合约

图 24: 部署Upvote合约

2.3 调用Upvote合约

2.3.1 创建2个新的账户

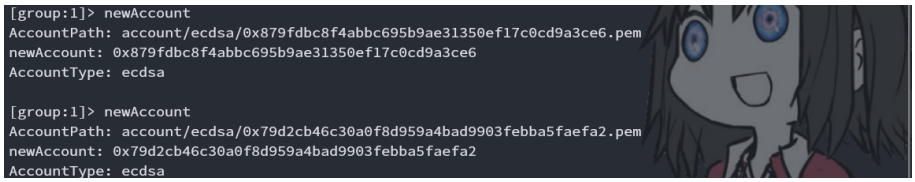

图 25: 创建2个新的账户

2.3.2 调用Upvote合约的接口

1. 使用当前新用户调用giveVotesToUser，失败

图 26: 使用当前新用户调用giveVotesToUser

2. 使用部署合约用户调用giveVotesToUser，成功，说明部署合约的用户是publisher。

图 27: 使用部署合约用户调用giveVotesToUser

3. publisher调用publish，如果遇到相同内容会报错

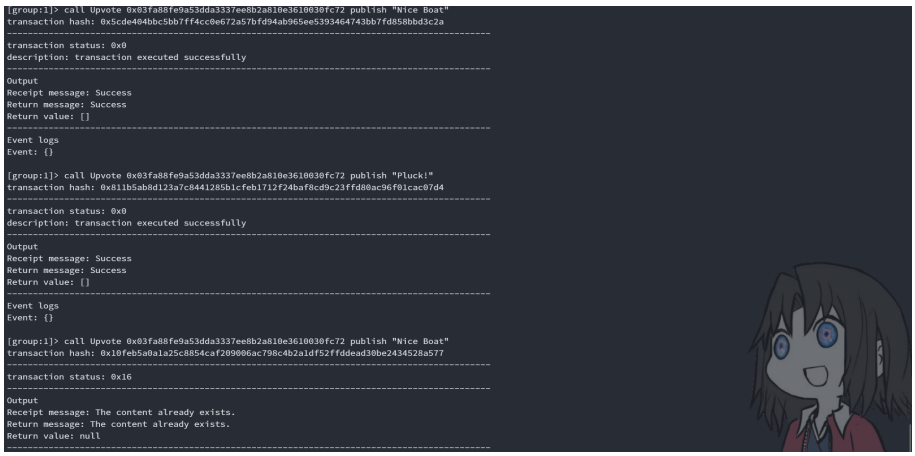

图 28: publisher调用publish

4. 切换用户调用upvote

- 当票数足够时，成功运行

图 29: 调用upvote票数足够

- 当票数不够时，报错

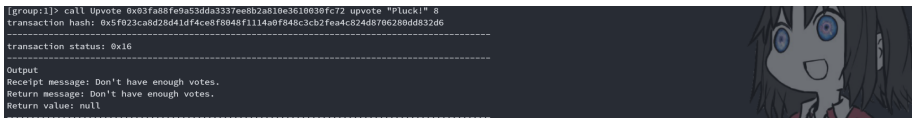

图 30: 调用upvote票数不够

5. 调用winner

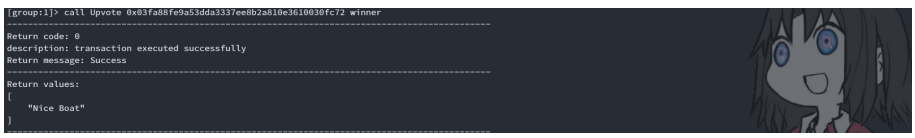

图 31: 调用winner

3 使用命令查看一个区块，并对各个字段进行解释。

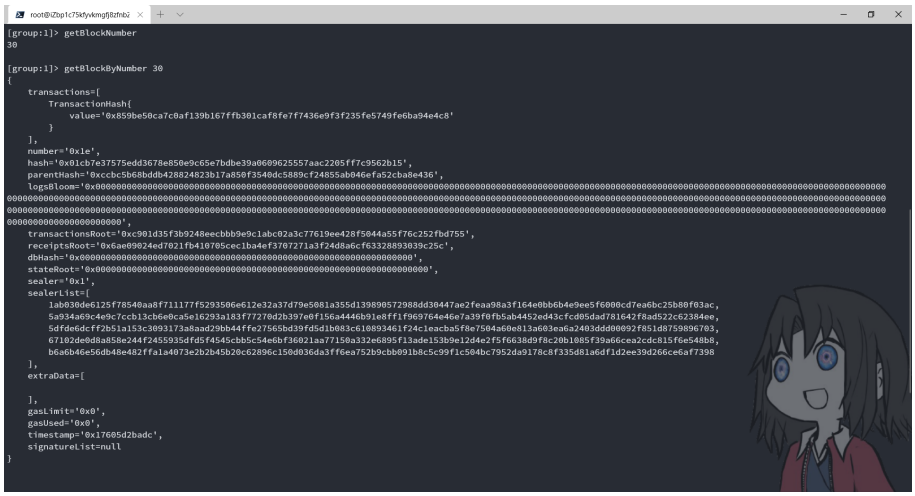

图 32: 调用upvote票数不够

- transactions：交易列表
  - TransactionHash：交易哈希
- number：区块高度
- hash：区块哈希
- parentHash：父区块哈希
- logsBloom：log的布隆过滤器值
- transactionRoot：区块内所有交易的merkle根
- receiptsRoot：区块内所有交易回执的merkle根
- dbHash：记录交易数据变更的哈希
- stateRoot：状态根哈希
- sealer：共识节点序号
- sealerList：共识节点列表
- extraData：交易内的extraData
- gasLimit：区块中允许的gas最大值
- gasUsed：区块中所有交易消耗的gas
- timestamp：时间戳，单位毫秒
- signatureList：PBFT共识的签名列表