区块链大作业：基于区块链的供应链金融平台

0. 实验前准备

① 参考文档

- FISCO BCOS 安装 官方文档
- 群组新增节点部分
- FISCO BCOS API

② 系统环境

- 笔者选用的系统环境是 Centos 7，下文是任务步骤的完成和截图

1. 作业内容

- 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。

2. 私有链搭建（单群组FISCO BCOS联盟链的搭建）

① 准备环境

- 笔者使用 CentOS 安装依赖 openssl，curl，它们是开发部署工具 build_chian.sh脚本的依赖

1）安装 Centos 依赖

```
sudo yum install -y openssl openssl-devel
```

```
[beilineili@localhost ~]$ sudo yum install -y openssl openssl-devel
已加载插件：fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.xtom.com.hk
 * extras: centos.01link.hk
 * updates: mirror.xtom.com.hk
软件包 1:openssl-1.0.2k-19.el7.x86_64 已安装并且是最新版本
```

2）创建操作目录，下载安装脚本

```
cd ~ && mkdir -p fisco && cd fisco
```

3）下载 build_chain.sh 脚本

```
curl -#LO https://gitee.com/FISCO-BCOS/FISCO-BCOS/raw/master/tools/build_chain.sh && chmod u+x build_chain.sh
```

```
[beilineili@localhost ~]$ cd ~ && mkdir -p fisco && cd fisco
[beilineili@localhost fisco]$ curl -#LO https://github.com/FISCO-BCOS/FISCO-BCOS/releases/download/v2.6.0/build_chain.sh && chmod u+x build_chain.sh
######################################################################## 100.0%
```

② 搭建单群组4节点联盟链

- 在fisco目录下执行下面的指令，生成一条单群组4节点的FISCO链。要确保机器的30300$_{30303}$，20200$_{20203}$，8545~8548端口没有被占用。

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

- 命令执行成功，输出 All completed

```
[beilineili@localhost 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/v2.6.0/fisco-bcos.tar.gz ...
######################################################################## 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      : /home/beilineili/fisco/nodes
[INFO] CA Path         : /home/beilineili/fisco/nodes/cert/
==============================================================
[INFO] Execute the download_console.sh script in directory named by IP to get FISCO-BCOS console.
e.g.   bash /home/beilineili/fisco/nodes/127.0.0.1/download_console.sh -f
==============================================================
[INFO] All completed. Files in /home/beilineili/fisco/nodes
```

③ 启动FISCO BCOS链

- 启动所有节点，启动成功

```
bash nodes/127.0.0.1/start_all.sh
```

- 输出下面内容的响应

```
[beilineili@localhost 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
 node0 start successfully
 node1 start successfully
 node3 start successfully
 node2 start successfully
```

④ 检查进程

- 检查进程是否启动

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

- 输出下列信息，进程数为 4

```
[beilineili@localhost fisco]$ ps -ef | grep -v grep | grep fisco-bcos
beiline+  4810     1  1 00:11 pts/0    00:00:01 /home/beilineili/fisco/nodes/127.0.0.1/node0/../fisco-bcos -c config.ini
beiline+  4812     1  1 00:11 pts/0    00:00:01 /home/beilineili/fisco/nodes/127.0.0.1/node3/../fisco-bcos -c config.ini
beiline+  4814     1  1 00:11 pts/0    00:00:01 /home/beilineili/fisco/nodes/127.0.0.1/node1/../fisco-bcos -c config.ini
beiline+  4816     1  1 00:11 pts/0    00:00:01 /home/beilineili/fisco/nodes/127.0.0.1/node2/../fisco-bcos -c config.ini
```

⑤ 检查日志输出

- 查看节点 node0 链接的节点数，从输出连接信息可看出 node0 与另外3个节点有链接

```
tail -f log/log* | grep connected
```

```
[beilineili@localhost fisco]$ tail -f nodes/127.0.0.1/node0/log/log*  | grep connected
info|2020-11-27 00:14:11.161039|[P2P][Service] heartBeat,connected count=3
info|2020-11-27 00:14:21.161426|[P2P][Service] heartBeat,connected count=3
info|2020-11-27 00:14:31.161797|[P2P][Service] heartBeat,connected count=3
```

- 执行下面指令，检查是否在共识，不停输出`++++Generating seal`，表示共识正常

```
tail -f log/log* | grep +++
```

```
[beilineili@localhost fisco]$ tail -f nodes/127.0.0.1/node0/log/log*  | grep +++
info|2020-11-27 00:15:47.527087|[g:1][CONSENSUS][SEALER]++++++++++++++++ Generating seal on,blkNum=1,tx=0,nodeIdx=3,hash=d8b6b076...
info|2020-11-27 00:15:51.552501|[g:1][CONSENSUS][SEALER]++++++++++++++++ Generating seal on,blkNum=1,tx=0,nodeIdx=3,hash=95076ec9...
info|2020-11-27 00:15:55.610270|[g:1][CONSENSUS][SEALER]++++++++++++++++ Generating seal on,blkNum=1,tx=0,nodeIdx=3,hash=dade9a6c...
info|2020-11-27 00:15:59.644403|[g:1][CONSENSUS][SEALER]++++++++++++++++ Generating seal on,blkNum=1,tx=0,nodeIdx=3,hash=998bceec...
```

3. 配置及使用控制台

- 在控制台链接 FISCO BCOS 节点，实现查询区块链状态、部署调用合约等功能，能够快速获取到所需要的信息

① 准备依赖

- 安装 java

```
#centos系统安装java
sudo yum install -y java java-devel
```

- 获取控制台并回到 fisco 目录（因为网络问题导致长时间无法下载，所以更换指令）

```
cd ~/fisco && curl -#LO https://gitee.com/FISCO-BCOS/console/raw/master/tools/download_console.sh
```

```
[beilineili@localhost fisco]$ cd ~/fisco && curl -#LO https://github.com/FISCO-BCOS/console/releases/download/v2.6.1/download_console.sh && bash download_console.sh
######################################################################## 100.0%
[INFO] Downloading console 2.7.0 from https://github.com/FISCO-BCOS/console/releases/download/v2.7.0/console.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   640  100   640    0     0    515      0  0:00:01  0:00:01 --:--:--   516
 63 38.1M   63 24.3M    0     0   164k      0  0:03:57  0:02:31  0:01:26  166k
curl: (28) Operation timed out after 150000 milliseconds with 25519694 out of 39997483 bytes received
[WARN] Download speed is too low, try https://osp-1257653870.cos.ap-guangzhou.myqcloud.com/FISCO-BCOS/console/releases/v2.7.0/console.tar.gz
######################################################################## 100.0%
[beilineili@localhost fisco]$ cd ~/fisco && curl -#LO https://gitee.com/FISCO-BCOS/console/raw/master/tools/download_console.sh
######################################################################## 100.0%
```


- 注意下载完后需要 bash（笔者因为网络问题导致长时间无法下载用了别的指令，结果忘记 bash 导致控制台无法启动）

```
bash download_console.sh
```

```
[beilineili@localhost fisco]$ bash download_console.sh
[INFO] Downloading console 2.7.0 from https://github.com/FISCO-BCOS/console/releases/download/v2.7.0/console.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   640  100   640    0     0    616      0  0:00:01  0:00:01 --:--:--   616
100 38.1M  100 38.1M    0     0  1264k      0  0:00:30  0:00:30 --:--:-- 1408k
[INFO] Download console successfully
[INFO] unzip console successfully
```

- 拷贝控制台配置文件
- 若节点未采用默认端口，请将文件中的20200替换成节点对应的channel端口。

```
cp -n console/conf/config-example.toml console/conf/config.toml
```

- 配置控制台证书

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

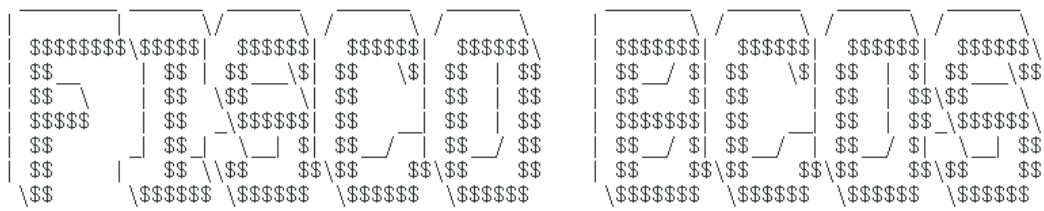
② 启动并使用控制台

- 启动，输出下述信息表明启动成功

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

```
[beilineili@localhost fisco]$ cp -n console/conf/config-example.toml console/conf/config.toml
[beilineili@localhost fisco]$ cp -r nodes/127.0.0.1/sdk/* console/conf/
[beilineili@localhost fisco]$ cd ~/fisco/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] >
```


- 用控制台获取信息，获取客户端版本

```
[group:1] > getNodeVersion
ClientVersion{
    version=2.7.0',
    supportedVersion=2.7.0',
    chainId=1',
    buildTime=20201126 08:00:45',
    buildType=Linux/clang/Release',
    gitBranch=HEAD',
    gitCommitHash=0bc47666979df4766723adf1ab9c5d80f5e40537'
}
```

4. 群组新增节点（新节点的加入）

- 下文内容主要参考FISCO BCOS 开发部署工具中 群组新增节点部分
- 接下来的操作，都在上一节生成的 `nodes/127.0.0.1` 目录下进行

① 为新节点生成私钥证书

- 获取证书生成脚本

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

```
[beilineili@localhost 127.0.0.1]$ curl -#LO https://gitee.com/FISCO-BCOS/FISCO-BCOS/raw/master/tools/gen_node_cert.sh
######################################################################## 100.0%
```

- 生成新节点私钥证书

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

```
[beilineili@localhost 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
```

② 准备配置文件

- 拷贝群组1中节点node0配置文件与工具脚本

```
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
- 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。假设新节点 IP:Port 为127.0.0.1:30304 则，修改后的 `[P2P]` 配置为: 如下图

```
[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
    ;enable_compress=true
    ; 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
    node.4=127.0.0.1:30304
```

- 启动新节点

```
sudo ./start.sh
```

```
beilineili@localhost newNode]$ sudo ./start.sh
sudo] beilineili 的密码：
newNode start successfully
```

- 通过console将新节点加入群组1
- `nodeID` 可以通过命令 `cat newNode/conf/node.nodeid` 来获取

```
[beilineili@localhost newNode]$ cat conf/node.nodeid
cca980607820e2e43255349061d190affbbale2208d29648cc6a89dbd289c5a8ca7c4df32cc16cfdb37849f04beaf3be48181378913f342b40f3ad166e54abb5
```

- 打开控制台添加共识节点

```
[group:1]> addSealer cca980607820e2e43255349061d190affbbale2208d29648cc6a89dbd289c5a8ca7c4df32cc16cfdb37849f04beaf3be48181378913f342b40f3ad166e54abb5
>
{
    "code":-51100,
    "msg":"Invalid node ID"
}
```

- 检查连接和共识（同上文命令）

```
ps -ef | grep -v grep | grep fisco-bcos
tail -f log/log* | grep connected
tail -f log/log* | grep +++
```

```
[beilineili@localhost newNode]$ ps -ef | grep -v grep | grep fisco-bcos
beiline+  4237     1  2 17:51 pts/0    00:02:13 /home/beilineili/fisco/nodes/127.0.0.1/node3/../fisco-bcos -c config.ini
beiline+  4239     1  2 17:51 pts/0    00:02:12 /home/beilineili/fisco/nodes/127.0.0.1/node0/../fisco-bcos -c config.ini
beiline+  4241     1  2 17:51 pts/0    00:02:12 /home/beilineili/fisco/nodes/127.0.0.1/node2/../fisco-bcos -c config.ini
beiline+  4243     1  2 17:51 pts/0    00:02:13 /home/beilineili/fisco/nodes/127.0.0.1/node1/../fisco-bcos -c config.ini
root      6242     1  0 19:30 pts/0    00:00:02 /home/beilineili/fisco/nodes/127.0.0.1/newNode/../fisco-bcos -c config.ini
[beilineili@localhost newNode]$ tail -f log/log* | grep connected
info|2020-11-27 19:36:20.643674|[P2P][Service] heartBeat,connected count=4
info|2020-11-27 19:36:30.643829|[P2P][Service] heartBeat,connected count=4
info|2020-11-27 19:36:40.644085|[P2P][Service] heartBeat,connected count=4
info|2020-11-27 19:36:50.644152|[P2P][Service] heartBeat,connected count=4
info|2020-11-27 19:37:00.644213|[P2P][Service] heartBeat,connected count=4
info|2020-11-27 19:37:10.644276|[P2P][Service] heartBeat,connected count=4
info|2020-11-27 19:37:20.644323|[P2P][Service] heartBeat,connected count=4
info|2020-11-27 19:37:30.644512|[P2P][Service] heartBeat,connected count=4
info|2020-11-27 19:37:40.644579|[P2P][Service] heartBeat,connected count=4
info|2020-11-27 19:37:50.644633|[P2P][Service] heartBeat,connected count=4
info|2020-11-27 19:38:00.644742|[P2P][Service] heartBeat,connected count=4
```


5. 智能合约（部署及调用HelloWorld合约）

- 打开控制台，获取节点信息

```
[group:1]> getPeers
[
    PeerInfo{
        nodeID=cca980607820e2e43255349061d190affbba1e2208d29648cc6a89dbd289c5a8ca7c4df32cc16cfdb37849f04beaf3be48181378913f342b40f3ad166e54abb5',
        iPAndPort=127.0.0.1:44160',
        node=node',
        agency=agency',
        topic=[

        ]'
    },
    PeerInfo{
        nodeID=204e2fce31991b27a46a0006e5e2df2ed7b8939d9d3438234bb0555a4d2765e40357cb06a98a2ea14d119184c7486b82bbe47ad2af5139cd0476155fb80f6f5d',
        iPAndPort=127.0.0.1:30303',
        node=node3',
        agency=agency',
        topic=[

        ]'
    },
    PeerInfo{
        nodeID=1df96b0f84101ff22be5e58c0e6f872476384c8d3903055ee082e516f966d1300dfd9e13cba44a2c5c4c863bd033aef3cb85a18f17bce3a6889a22c0c3ebbf3e',
        iPAndPort=127.0.0.1:44122',
        node=node2',
        agency=agency',
        topic=[

        ]'
    },
    PeerInfo{
        nodeID=f547180f70aaf896e359aa4cba184020d46e7a5c22982635f06d20f83c9eb232501b03a3b0903eb197dce15ca8fc87dff258382cf84ba2603ad7b3fdacaef35f',
        iPAndPort=127.0.0.1:30300',
        node=node0',
        agency=agency',
        topic=[
            _block_notify_1
        ]'
    }
]
```


① 编写HelloWorld合约

- HelloWorld 合约已经内置于控制台中，位于控制台目录下 contracts/solidity/HelloWorld.sol
- 合约内容如下，它提供了两个接口，即 `get` 和 `set`，获得和设置合约变量 `name`

```
pragma solidity>=0.4.24 <0.6.11;

contract HelloWorld {
    string name;

    constructor() public {
        name = "Hello, World!";
    }

    function get() public view returns (string memory) {
        return name;
    }

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

② 部署HelloWorld合约

- 参考下面命令部署
- 在控制台输入以下指令 部署成功则返回合约地址

```
[group:1] > deploy HelloWorld
transaction hash: 0x2811963f6a0af85a008ff8196be2db6bf0d1acb1294de3cc2d8194a5fa268e69
contract address: 0x6f7c8d3896762bb8ff15f15fb431a1de3ed3b2bc
```

- 合约地址即为 0x6f7c8d3896762bb8ff15f15fb431a1de3ed3b2bc

③ 调用HelloWorld合约

```
# 查看当前块高
[group:1]> getBlockNumber

# 调用get接口获取name变量 此处的合约地址是deploy指令返回的地址
[group:1]> call HelloWorld address get

# 查看当前块高，块高不变，因为get接口不更改账本状态
[group:1]> getBlockNumber

# 调用set设置name
[group:1]> call HelloWorld address set

# 再次查看当前块高，块高增加表示已出块，账本状态已更改
[group:1]> getBlockNumber

# 调用get接口获取name变量，检查设置是否生效
[group:1]> call HelloWorld address get

# 退出控制台
[group:1]> quit
```

- 命令执行结果如下

```
[group:1] > getBlockNumber
1
```

```
[group:1] > call HelloWorld 0x6f7c8d3896762bb8ff15f15fb431a1de3ed3b2bc get
-------------------------------------------------------------------------------
Return code: 0
description: transaction executed successfully
Return message: Success
-------------------------------------------------------------------------------
Return values:
[
    "Hello,World!"
]
-------------------------------------------------------------------------------
```

```
[group:1] > getBlockNumber
1

[group:1] > call HelloWorld 0x6f7c8d3896762bb8ff15f15fb431a1de3ed3b2bc set "FISCO BCOS"
transaction hash: 0xa668e742173e4616cfc26c4389684af7bc81f9950608655f9ed6e58701e7dd6f
---------------------------------------------------------------------------------------
transaction status: 0x0
description: transaction executed successfully
---------------------------------------------------------------------------------------
Output
Receipt message: Success
Return message: Success
Return value: []
---------------------------------------------------------------------------------------
Event logs
Event: {}
```



```
[group:1] > getBlockNumber
2

[group:1] > call HelloWorld 0x6f7c8d3896762bb8ff15f15fb431a1de3ed3b2bc get
---------------------------------------------------------------------------------------
Return code: 0
description: transaction executed successfully
Return message: Success
---------------------------------------------------------------------------------------
Return values:
[
    "FISCO BCOS"
]
---------------------------------------------------------------------------------------

group:1] > quit
beilineili@localhost console] $
```


6. 查看区块

①阅读 FISCO BCOS API

- 从 FISCO BCOS API 得知可使用 getBlockByNumber 返回根据区块高度查询的区块信息，参数和返回值如下：

getBlockByNumber

返回根据区块高度查询的区块信息

参数

- `groupID`: `unsigned int` - 群组ID
- `blockNumber`: `string` - 区块高度(十进制字符串或0x开头的十六进制字符串)
- `includeTransactions`: `bool` - 包含交易标志(true显示交易详细信息，false仅显示交易的hash)

返回值

- `object` - 区块信息，字段如下：
  - `dbHash`: `string` - 记录交易数据变更的哈希
  - `extraData`: `array` - 附加数据
  - `gasLimit`: `string` - 区块中允许的gas最大值
  - `gasUsed`: `string` - 区块中所有交易消耗的gas
  - `hash`: `string` - 区块哈希
  - `logsBloom`: `string` - log的布隆过滤器值
  - `number`: `string` - 区块高度
  - `parentHash`: `string` - 父区块哈希
  - `receiptsRoot`: `string` - 区块内所有交易回执的merkle根
  - `sealer`: `string` - 共识节点序号
  - `sealerList`: `array` - 共识节点列表
  - `signatureList`: `string` - PBFT共识的签名列表
  - `stateRoot`: `string` - 状态根哈希
  - `timestamp`: `string` - 时间戳，单位毫秒
  - `transactions`: `array` - 交易列表，当 `includeTransactions` 为 `false` 时，显示交易的哈希。当 `includeTransactions` 为 `true` 时，显示交易详细信息（详细字段见getTransactionByHash）
  - `transactionsRoot`: `string` - 区块内所有交易的merkle根

② 运行 getBlockByNumber 查询区块信息

```
[group:1]> getBlockByNumber 4
{
    transactions=[
        TransactionHash{
            value='0x2811963f6a0af85a008ff8196be2db6bf0d1acb1294de3cc2d8194a5fa268e69'
        }
    ],
    number='0x4',
    hash='0x4b377e091c0c0b2bc319296eea626a2dffe28c87b8f8ee6e53030560896ab6ed',
    parentHash='0x4d266be15bb95df6314f04d4bb04cb60266e86891b66395bb2412459afd167ad',
    logsBloom='0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000',
    transactionsRoot='0xfd4ef2d96fc803501cc67c3726851d3151ddde41317f785d5a89867467a891b7',
    receiptsRoot='0x84cb2a57ab99f5c9391c39a2b9ac80707fd3ef56e70d3abf64df3e8a237ae4b3',
    dbHash='0xd315858b29e88b3aa4db11d146c66ae3c6d68a93f4e4bd9969766ec3207f8b17',
    stateRoot='0xd315858b29e88b3aa4db11d146c66ae3c6d68a93f4e4bd9969766ec3207f8b17',
    sealer='0x1',
    sealerList=[
        1df96b0f84101ff22be5e58c0e6f872476384c8d3903055ee082e516f966d1300dfd9e13cba44a2c5c4c863bd033aef3cb85a18f17bce3a6889a22c0c3ebbf3e,
        204e2fce31991b27a46a0006e5e2df2ed7b8939d9d3438234bb0555a4d2765e40357cb06a98a2ea14d119184c7486b82bbe47ad2af5139cd0476155fb80f6f5d,
        475a87765ae992bafbb0780b92dddb815c5e8b28fa91158c394f3cadb7729e6c3e1a62069ad78a317ac3fdfe161d8fdd45ae56cf26567dbbd79949f8268cbfb0,
        f547180f70aaf896e359aa4cba184020d46e7a5c22982635f06d20f83c9eb232501b03a3b0903eb197dce15ca8fc87dff258382cf84ba2603ad7b3fdacaef35f
    ],
    extraData=[

    ],
    gasLimit='0x0',
    gasUsed='0x0',
    timestamp='0x17609fd29c2',
    signatureList=null
}
```


- 根据 API 里对参数和返回值的解释，我们可知各个字段的含义：
- transactions：转移值

```
transactions=[
    TransactionHash{   转移哈希
        value='0x2811963f6a0af85a008ff8196be2db6bf0d1acb1294de3cc2d8194a5fa268e69'
    }
],
```

- number：区块高度

```
number='0x4',
```

- hash：区块哈希

```
hash='0x4b377e091c0c0b2bc319296eea626a2dffe28c87b8f8ee6e53030560896ab6ed',
```

- parentHash：父区块哈希

```
parentHash='0x4d266be15bb95df6314f04d4bb04cb60266e86891b66395bb2412459afd167ad',
```

- logsBloom：log的布隆过滤器值

```
logsBloom='0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000',
```

- transactionRoot：区块内所有交易的 merkle 根

```
transactionsRoot='0xfd4ef2d96fc803501cc67c3726851d3151ddde41317f785d5a89867467a891b7',
```

- receiptsRoot：区块内所有交易回执的 merkle 根

```
receiptsRoot='0x84cb2a57ab99f5c9391c39a2b9ac80707fd3ef56e70d3abf64df3e8a237ae4b3',
```

- dbHash：记录交易数据变更的哈希

```
stateRoot='0xd315858b29e88b3aa4db11d146c66ae3c6d68a93f4e4bd9969766ec3207f8b17',
```

- stateRoot：状态根哈希

```
stateRoot='0xd315858b29e88b3aa4db11d146c66ae3c6d68a93f4e4bd9969766ec3207f8b17',
```

- sealer：共识节点序号

```
sealer='0x1',
```

- sealerList：共识节点列表

```
sealerList=[
    1df96b0f84101ff22be5e58c0e6f872476384c8d3903055ee082e516f966d1300dfd9e13cba44a2c5c4c863bd033aef3cb85a18f17bce3a6889a22c0c3ebbf3e,
    204e2fce31991b27a46a0006e5e2df2ed7b8939d9d3438234bb0555a4d2765e40357cb06a98a2ea14d119184c7486b82bbe47ad2af5139cd0476155fb80f6f5d,
    475a87765ae992bafbb0780b92dddb815c5e8b28fa91158c394f3cadb7729e6c3e1a62069ad78a317ac3fdfe161d8fdd45ae56cf26567dbbd79949f8268cbfb0,
    f547180f70aaf896e359aa4cba184020d46e7a5c22982635f06d20f83c9eb232501b03a3b0903eb197dce15ca8fc87dff258382cf84ba2603ad7b3fdacaef35f
],
```

- extraData：附加数据

```
extraData=[

],
```

- gasLimit：区块中允许的 gas 最大值

```
gasLimit='0x0',
```

- gasUsed：区块中所有交易消耗的gas

```
gasUsed='0x0',
```

- timestamp：时间戳，单位毫秒

```
timestamp='0x17609fd29c2',
```

- signatureList：PBFT共识的签名列表

```
signatureList=null
```