|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 修改人 | 时间 | 原因 |
| V1.0.0 | 郭 旭 | 2018/4/27 | 创建 增加IBFT单机多节点和多机部署方法.  增加IBFT web3.js APIs使用参考. |
| V1.0.1 | 郭 旭 | 2018/5/7 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 基于Istanbul BFT的部署实践

**说明:**

此部分实践了两种采用Istanbul BFT共识算法的部署方案, 第一种方案验证了在单机上运行多个lanchain node的部署方法; 第二种方案验证了多台虚拟机互联,每个虚拟机上只运行一个lanchain node的部署方法.

此部署方案采用static nodes的方式进行节点配置.

## 1.1 单机多节点部署方法

### 1.1.1网络拓扑

此部署实践中共有5个节点组成私有链,节点信息如下:

Node1: 端口 40001, 数据文件夹: node1, 作为挖矿节点.

Node2: 端口 40002, 数据文件夹: node2, 作为挖矿节点.

Node3: 端口 40003, 数据文件夹: node3, 作为挖矿节点.

Node4: 端口 40004, 数据文件夹: node4, 作为挖矿节点.

Node5: 端口 40005, 数据文件夹: node5, 作为交易节点.

Node5 (交易节点)

Node1(挖矿节点)

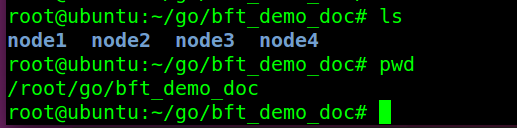
Node2(挖矿节点)

Node4(挖矿节点)

Node3(挖矿节点)

### 1.1.2准备工作

#### 1.1.2.1为4个挖矿节点创建数据文件夹



#### 1.1.2.2交易账号准备

**创建4个账户用于后续交易操作:**

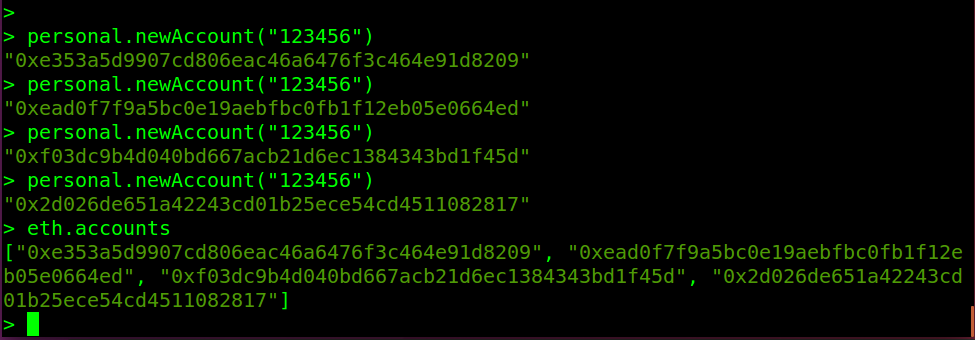
* 1. 执行如下命令进入node1的控制台

gwan–datadir “./”–nodiscover console

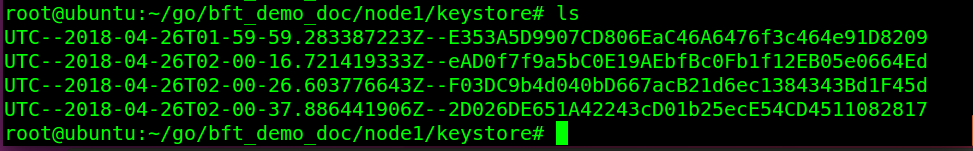


* 1. 创建4个账号

Personal.newAccount(“123456”)



* 1. 创建账号之后账号对应的私钥文件会生成在 node1/keystore目录下(keystore存储在哪个目录决定于步骤1中—datadir参数的设置).



#### 1.1.2.3 挖矿节点列表配置

* 1. 进入到各个节点的数据文件夹,执行bootnode命令生成nodekey文件.

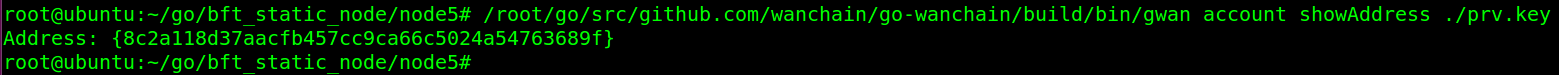
./bootnode -genkey prv.key //生成节点的私钥文件prv.key

(所有需要挖矿的节点都需要执行此步骤以生成每个节点各自的私钥文件.)

* 1. 获取nodekey对应的address.

对每一个步骤1中生成的私钥文件都需要执行如下命令来获得地址:

gwan account showAddress ./prv.key



(showAddress使用方法参见备注.)

目前使用的测试节点nodekey对应的address如下:

"0x4B9ED090E635CCE279FE5BD7948F93EDEBB9E634",

"0x0f0b004b1fa88eb885091df840316a56fcb80df4",

"0x9d14e190712008129a440473c010b3fd7599153a",

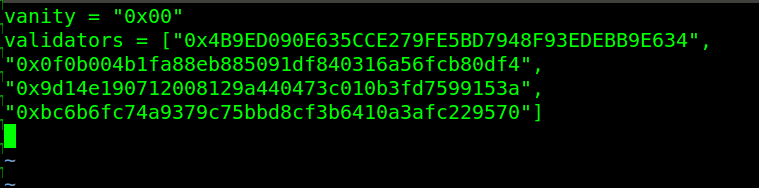
"0xbc6b6fc74a9379c75bbd8cf3b6410a3afc229570"

(每个地址对应一个拓扑环境中节点的nodekey)

* 1. 生成toml文件

Toml文件供后续步骤使用,用来生成extraData.

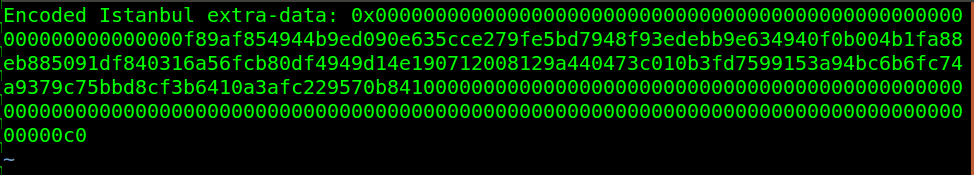
Toml文件格式如下(validators中为步骤2生成的nodekey地址):



* 1. 通过istanbul\_tools生成用于 genesis.json的 extraData. (Istanbul\_tools使用方法参考”备注”节)

/root/go/src/github.com/getamis/istanbul-tools/build/bin/istanbul extra encode --config ./bft.toml > address.gx

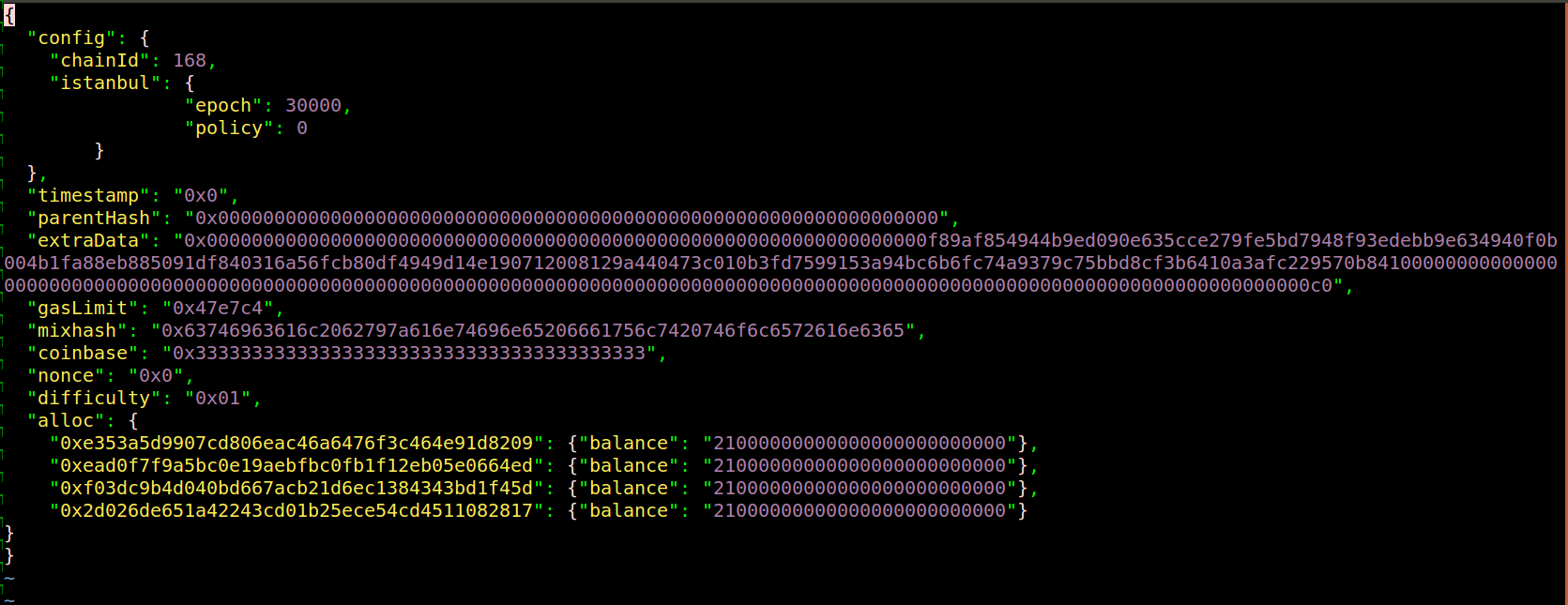
生成的address.gx内容如下:



其中”Encoded Istanbul extra-data:”后面的内容为经过RLP编码的validators,需要拷贝到genesis.json的extraData字段.

* 1. 把生成的extraData更新到genesis.json的extraData字段中

更新后的genesis.json内容如下:



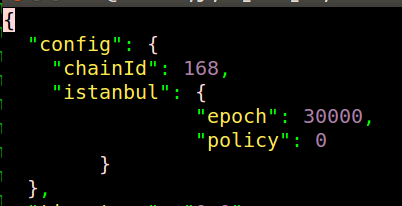
* 1. 把交易账户更新到genesis.json中.

(alloc字段为更新之后的交易账户信息,需要与之前步骤中的修改保持一致.)

#### 1.1.2.4 共识算法配置

使用istanbul共识算法,需要在genesis.json文件中进行配置.

配置内容如下:

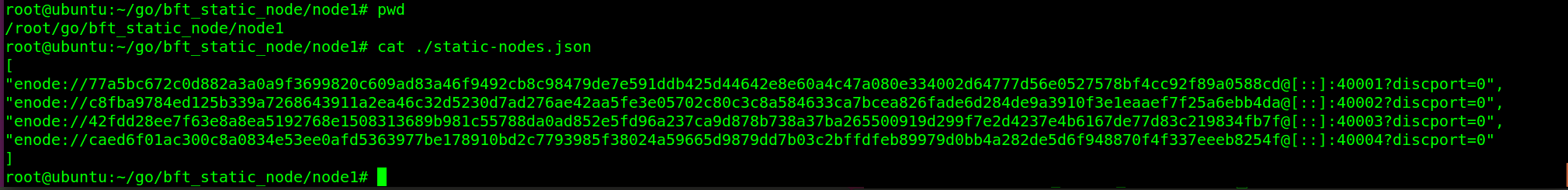


(至此, 环境准备完毕.)

### 1.2.3私链配置与运行

1. 通过static node的方式连接需要挖矿的各个节点.

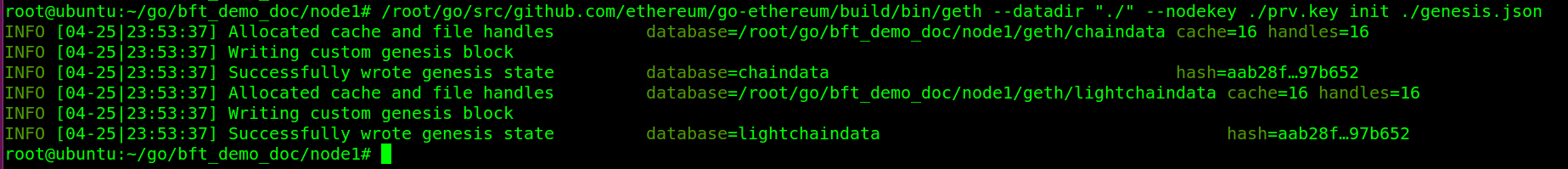
把需要挖矿的节点的enode按照图片中的格式写到文件 static-nodes.json中(lanchain通过检查数据文件夹中以”static-nodes.json”命令的文件是否存在来判断是否通过static node的方式连接peers,因此文件名字不能发生错误).



(所有挖矿的节点均需要拷贝此static-nodes.json文件到自己的数据文件夹.)

1. 节点初始化(各个节点均执行如下命令)
   1. 为了保持测试环境干净,需要进入各个节点的数据文件夹(例如:node<n>)删除各个节点数据文件夹中的gwan文件夹, 删除ipc文件及history文件.
   2. 在各个节点的node<n>文件夹下均执行如下命令.

/root/go/src/github.com/wanchain/go-wanchain/build/bin/gwan --datadir "./" --nodekey ./prv.key init ./genesis.json



1. 节点启动 (各个节点均执行如下命令)

/root/go/src/github.com/wanchain/go-wanchain/build/bin/gwan --datadir "./" --nodekey ./prv.key --networkid 168 --port 40001 --nodiscover console (注意每个节点启动的时候使用的端口号不相同.)

1. 查看节点的peers.

在每个节点的javascript console中执行 net.peerCount命令来查看peers数,此拓扑环境中,各个节点的peers均应该为3.

1. 添加peers完成之后每个节点执行命令miner.start()开始挖矿.
2. 拷贝node4文件夹为node5, 删除文件夹里的gwan, history, prv.key及static-nodes.json文件.
   1. 初始化节点(因为不参与挖矿所以不需要带有nodekey参数)

/root/go/src/github.com/wanchain/go-wanchain/build/bin/gwan --datadir "./" init ./genesis.json

* 1. 运行节点(不需要带有nodekey参数)

/root/go/src/github.com/wanchain/go-wanchain/build/bin/gwan --datadir "./" --networkid 168 --port 40005 --nodiscover console

* 1. 在node5的console上执行 admin.nodeInfo.enode 查看node5的enode.
  2. 在node1的console中执行 admin.addPeer(<node5 enode>)把node5加入到p2p网络中.

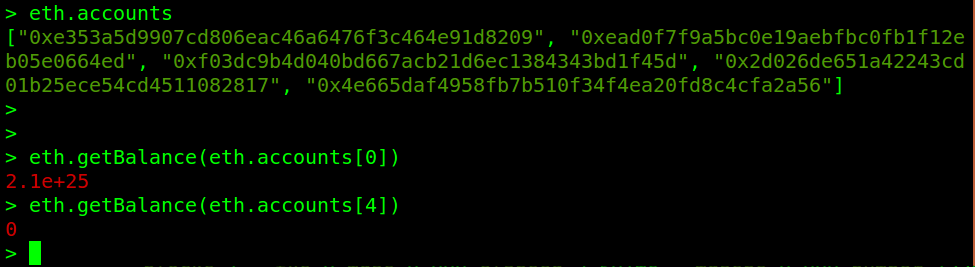
**注意:**

因为node5的nodekey address并没有被配置在istanbul miner list中,所以node5没有权限参与挖矿.

如果使用node5进行挖矿会打印如下信息:

****

1. 创建一个新的账户,然后查询之前配置的有余额的账户钱数正确, 新的账户中钱数为0.
2. 从一个有余额的账户给新账户转账

****

1. 解锁第0个账户

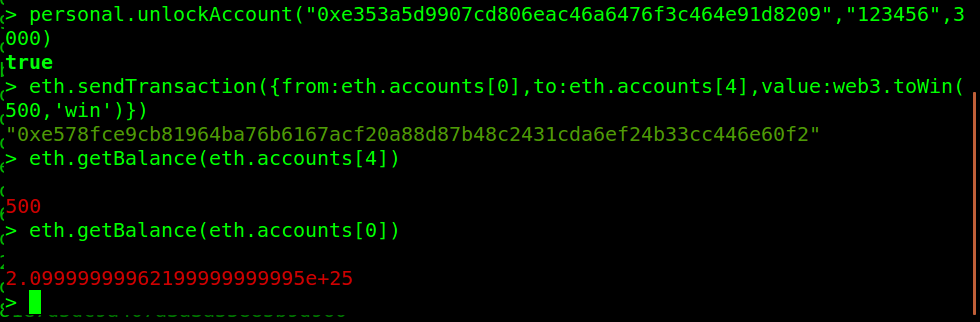
personal.unlockAccount("0xe353a5d9907cd806eac46a6476f3c464e91d8209","123456",3000)

1. 从第0个账户发送500个win给第4个账户,即新创建的账户

> eth.sendTransaction({from:eth.accounts[0],to:eth.accounts[4],value:web3.toWin(500,'win')})

"0xe578fce9cb81964ba76b6167acf20a88d87b48c2431cda6ef24b33cc446e60f2"

1. 执行交易之后查询余额

****

此步骤正确执行之后会发生余额的转移, 如果余额符合预期说明代码中istanbul consensus及测试环境配置正确.

1. 查看交易被打包到哪个块中

****

从命令结果来看,交易被正确打包到了区块中,从而验证了istanbul consenses共识算法运行正确及环境配置过程正确.

## 1.2多机器部署方法

### 1.2.1 网络拓扑

192.168.169.135:40003

挖矿/交易

192.168.160.134:40002  
挖矿/交易

192.168.160.133:40001  
挖矿/交易

### 1.2.2 准备工作

三个虚拟机，配置分别为：

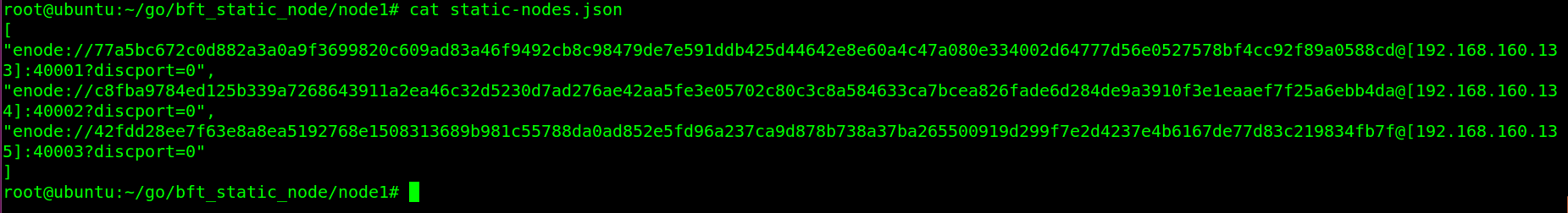
|  |  |  |  |
| --- | --- | --- | --- |
| ip地址 | 端口 | 数据文件夹 | 具备功能 |
| 192.168.160.133 | 40001 | Node1 | 挖矿、交易 |
| 192.168.160.134 | 40002 | Node2 | 挖矿、交易 |
| 192.168.160.135 | 40003 | Node3 | 挖矿、交易 |

其余准备工作参考单机多节点实践。

### 1.2.3 私链配置与运行

按照1.2.3节步骤一中说明生成static-nodes.json文件，static-nodes.json中每个enode的[::]部分需要替换成对应的ip地址。

生成后的文件内容如下：



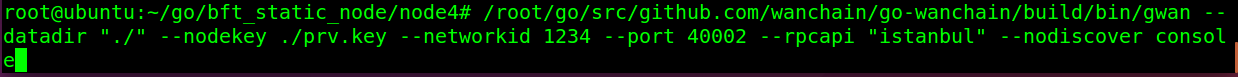
把制作好的static-nodes.json文件分别拷贝到192.168.160.133的node1文件夹，192.168.160.134的node2文件及192.168.160.135的node3文件夹各一份。

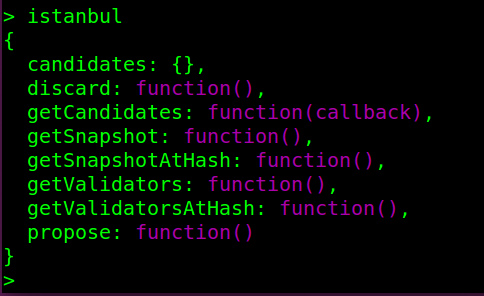
其他步骤与验证方法与1.2.3节相同。

## 1.3 istanbul web3.js扩展API使用示例

### 1.3.1 使能istanbul web3.js扩展

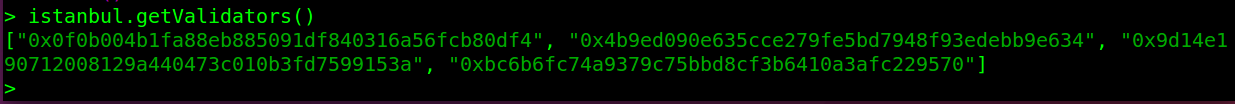
在启动gwan控制台的时候使用参数 –rpcapi “istanbul”.



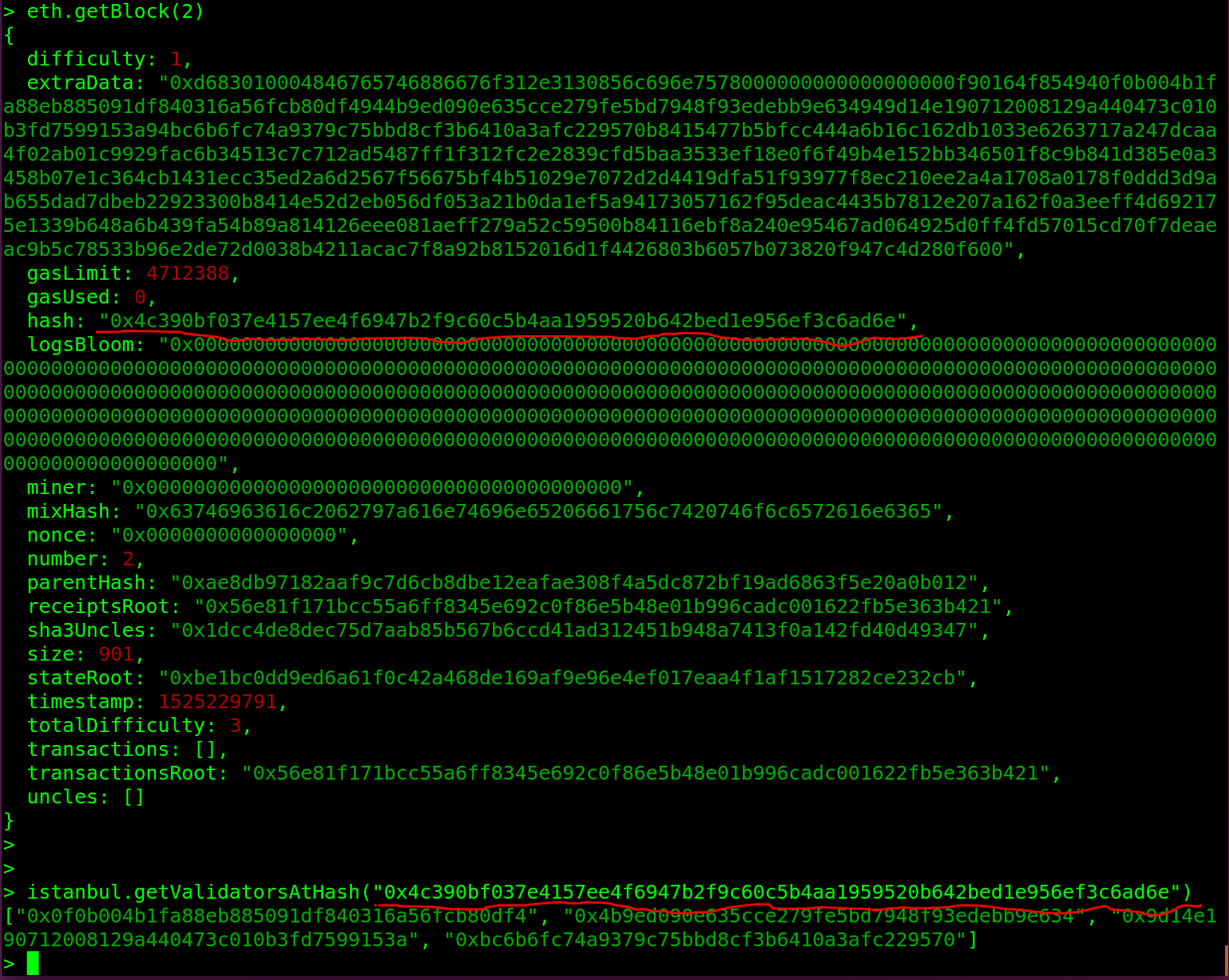


### 1.3.2 查看validators

Istanbul.getValidators():

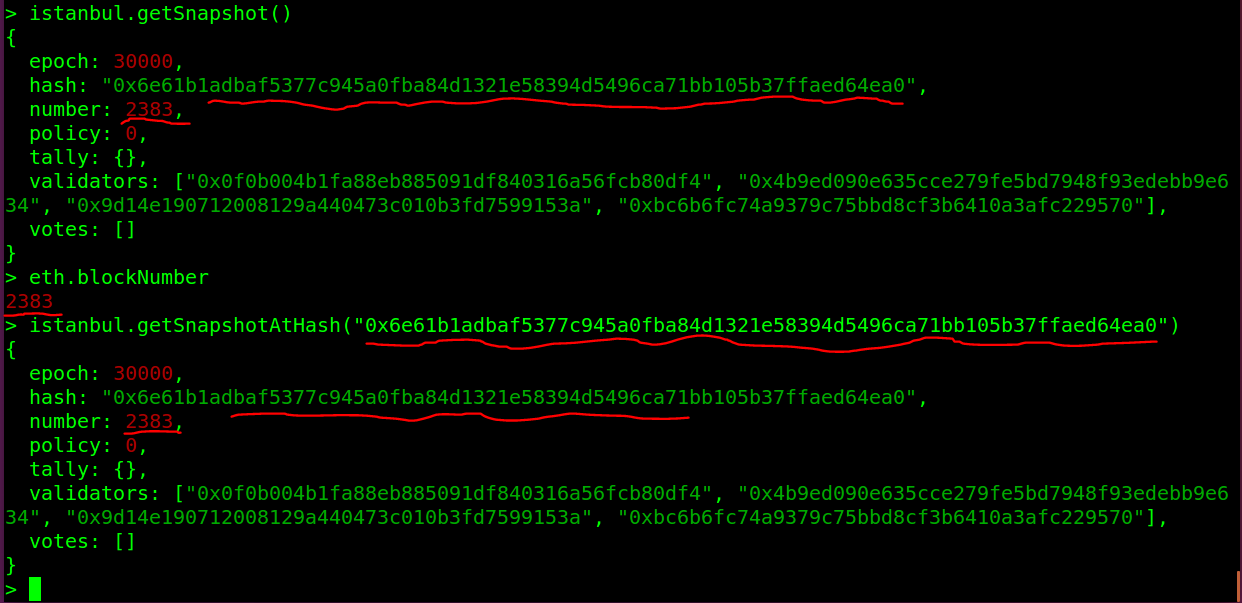


Istanbul.getValidatorsAtHash(<block hash>):



### 1.3.3 查看snapshot

Istanbul.getSnapshot(), Istanbul.getSnapshotAtHash(<block hash>):



### 1.3.4 动态添加/删除validators

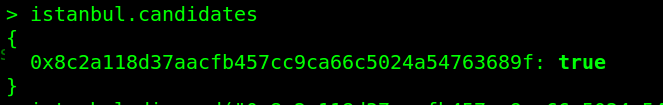


动态增加validators.

当超过1/2的validators投票之后, 被投票的validator会从validator列表中添加/删除.

Propose()命令对固化在genesis.json的validators与动态添加的validators均有效.

### 1.3.5 查看正被表决的validator



### 1.3.6 istanbul web3.js external APIs

<https://github.com/getamis/go-ethereum/wiki/RPC-API>

# 备注

## Istanbul tool

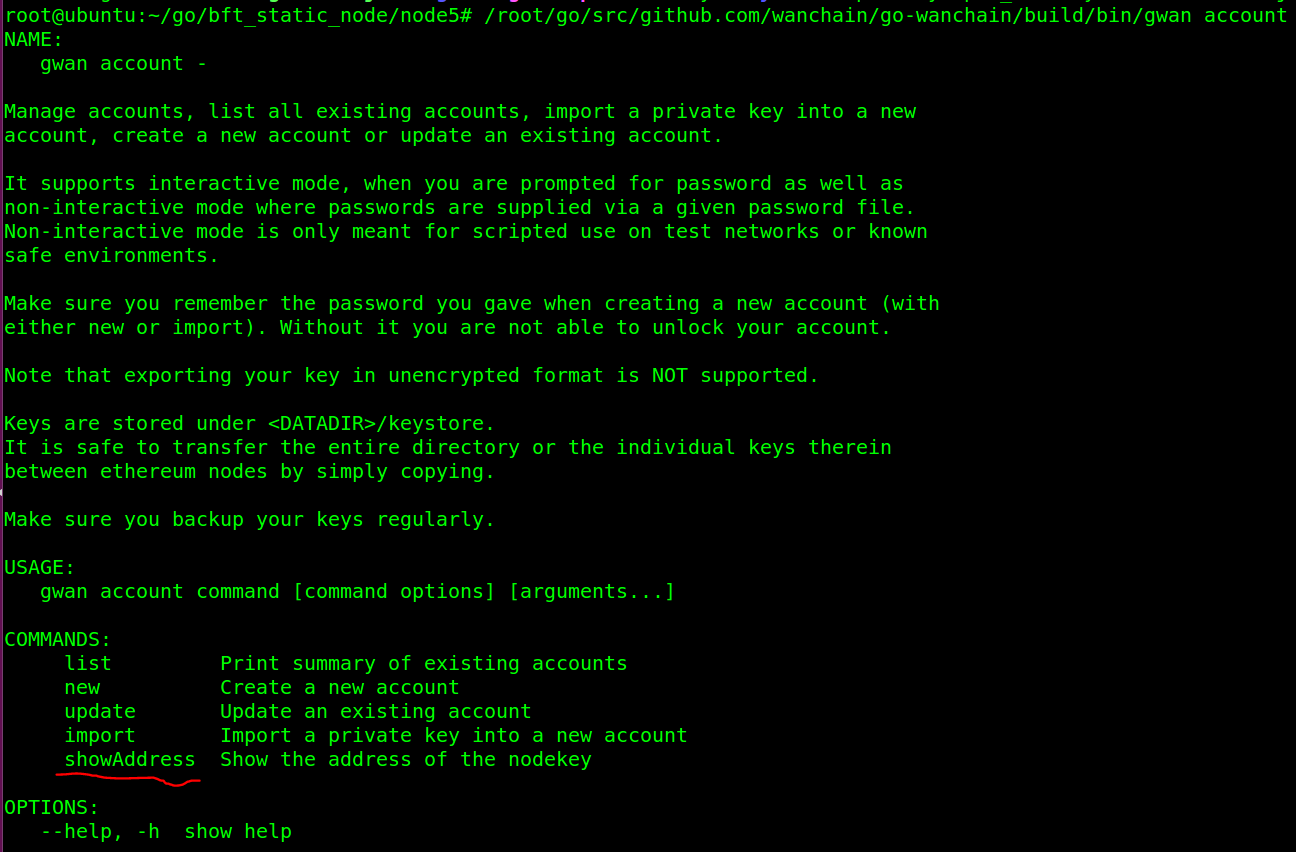
Istanbul tool源代码链接:

<https://github.com/getamis/istanbul-tools.git>

编译使用方法请参考此工具的readme.

## 获得nodekey的address

显示account命令帮助信息:



显示showAddress命令帮助信息:



---- END ----