|  |  |  |  |
| --- | --- | --- | --- |
| 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 | 修改: 1. 所有示例改为4个节点.  2. 示例中去掉了gwan参数 --nodiscover.  3. 通过gwan命令行参数生成账户. |
| V1.0.2 | 郭 旭 | 2018/5/27 | 修改: 1. 细化各步骤说明. |
|  |  |  |  |
|  |  |  |  |

# 基于Istanbul BFT的lanchain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节点的部署方法; 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 节点的部署方法.

此部署方案采用静态节点列表的方式进行节点之间的p2p连接配置.

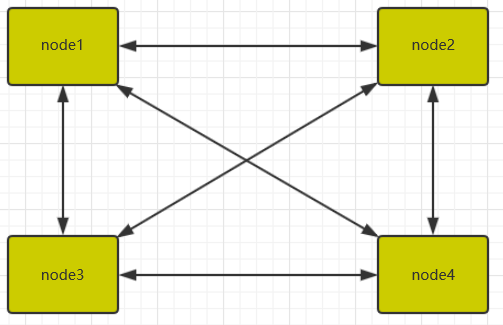
## 1.1 单机多节点部署方法

### 1.1.1网络拓扑

此部署实践中共有4个节点参与IBFT共识(挖矿),节点信息如下:

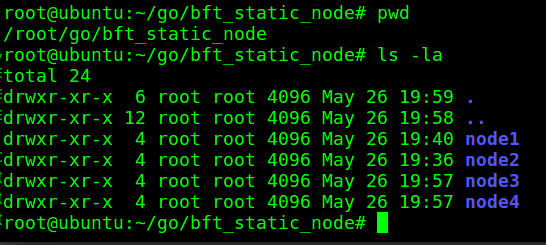
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **节点名称** | **数据文件路径** | **节点端口号** | **rpc端口号** | **rpc地址** | **网络ID** |
| node1 | /root/go/bft\_static\_node/node1 | 40001 | 50001 | 192.168.138.139 | 168 |
| node2 | /root/go/bft\_static\_node/node2 | 40002 | 50002 | 192.168.138.139 | 168 |
| node3 | /root/go/bft\_static\_node/node3 | 40003 | 50003 | 192.168.138.139 | 168 |
| node4 | /root/go/bft\_static\_node/node4 | 40004 | 50004 | 192.168.138.139 | 168 |

拓扑关系如下:



### 1.1.2环境配置

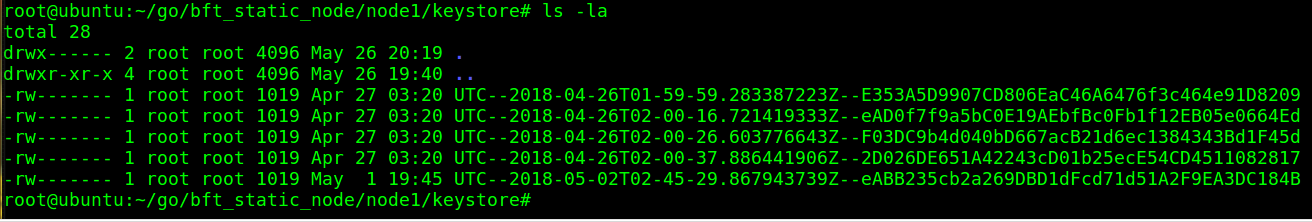
#### 1.1.2.1为4个挖矿节点创建数据文件夹



#### 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. 后续实践步骤中会通过node1节点发送交易,所以此处在node1节点的数据路径中创建用于交易的4个账号. 账号创建方法参见本文中"备注-账号创建"章节.
  2. 创建完成之后会在节点node1的数据文件夹生成4个账户,如下图.

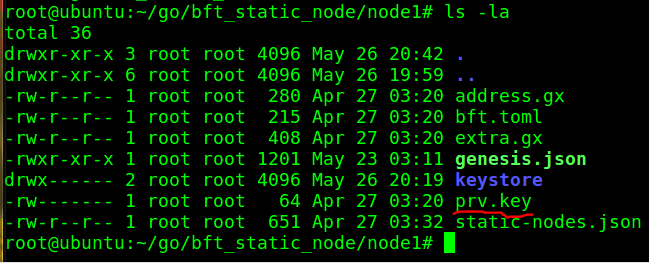


#### 生成节点私钥

进入到各节点的数据文件夹,执行bootnode命令生成nodekey文件.

bootnode -genkey prv.key

以节点node1为例, 生成私钥如下:



#### IBFT创世块文件配置

IBFT算法在创世块文件genesis.json中有其特殊配置,下面逐一说明:

1. 配置IBFT共识(挖矿)节点列表.

参见备注-IBFT共识节点列表配置.

1. 配置交易账户.

把在1.1.2.2章节生成的4个账户配置到创世块节点文件中.

配置方法参见备注-IBFT交易账户配置

1. 配置IBFT属性.

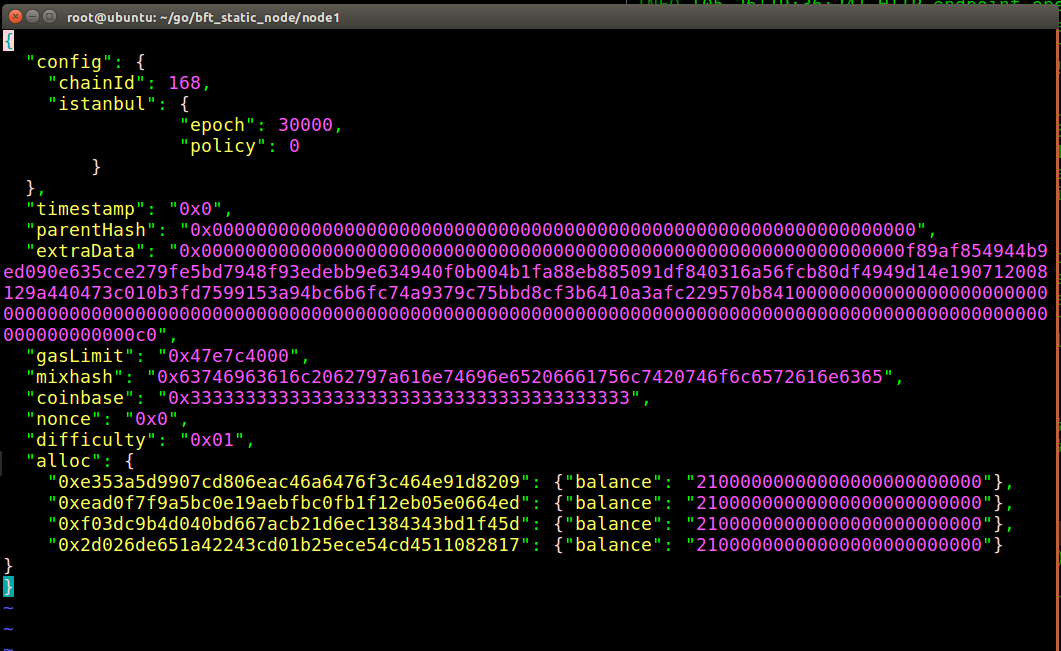
参见备注-IBFT属性配置.

1. 配置IBFT mixhash

使用IBFT必须配置mixhash为IBFT指定值.

配置方法参见备注-配置IBFT mixhash.

配置完成后的genesis.json文件应如下图所示:



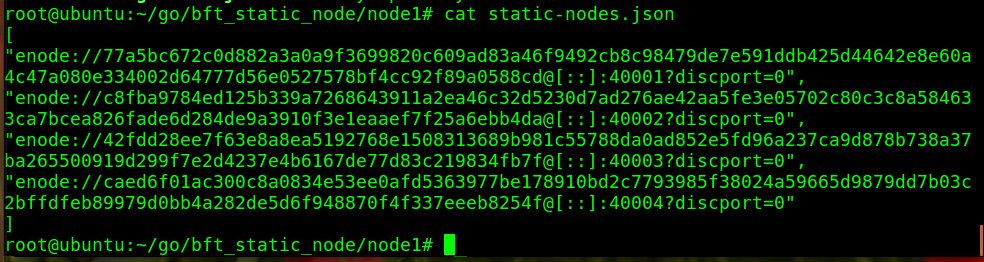
#### 静态节点列表生成

* 1. 获得各节点的enode.

获得enode的方法参见备注-获得节点enode章节.

* 1. 生成静态节点列表文件.

使用vim等编辑软件生成文件static-nodes.json, 把各节点的enode按照下图中的格式写入文件,保存退出. (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)



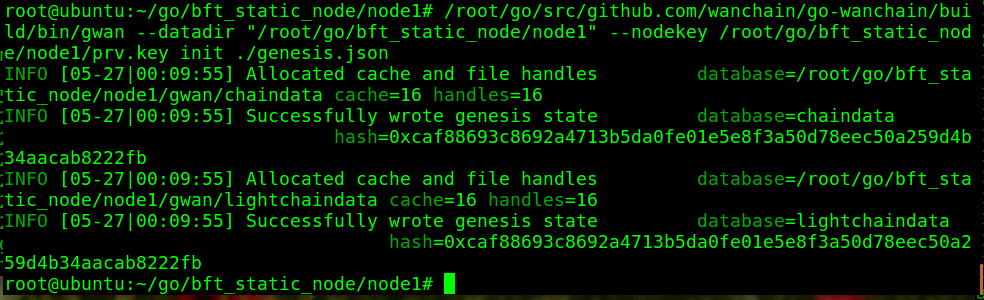
3. 拷贝static-nodes.json文件到各节点的数据文件夹.

### 1.1.3私链配置与运行

1. 节点初始化

对4个节点均执行如下命令进行节点初始化(需注意命令中的datadir及prv.key要使用对应节点中的文件.).

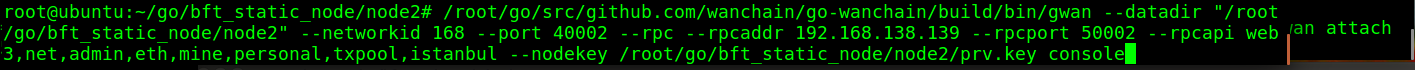
/root/go/src/github.com/wanchain/go-wanchain/build/bin/gwan --datadir "/root/go/bft\_static\_node/node1" --nodekey /root/go/bft\_static\_node/node1/prv.key init ./genesis.json



1. 各节点启动

对各节点执行如下启动命令,需要注意datadir, prv.key需要使用各节点对应文件.

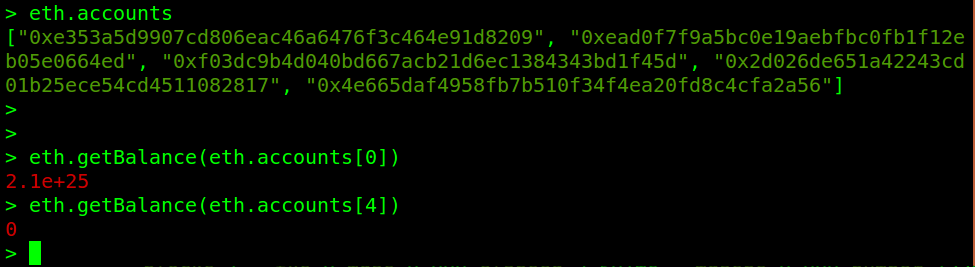
/root/go/src/github.com/wanchain/go-wanchain/build/bin/gwan --datadir "/root/go/bft\_static\_node/node2" --networkid 168 --port 40002 --rpc --rpcaddr 192.168.138.139 --rpcport 50002 --rpcapi web3,net,admin,eth,mine,personal,txpool,istanbul --nodekey /root/go/bft\_static\_node/node2/prv.key console



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,此拓扑环境配置中,各个节点的peers均应该为3.

1. 各节点执行命令miner.start()开始挖矿.
2. 进入到node1节点的数据文件夹node1,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.



1. 解锁第0个账户

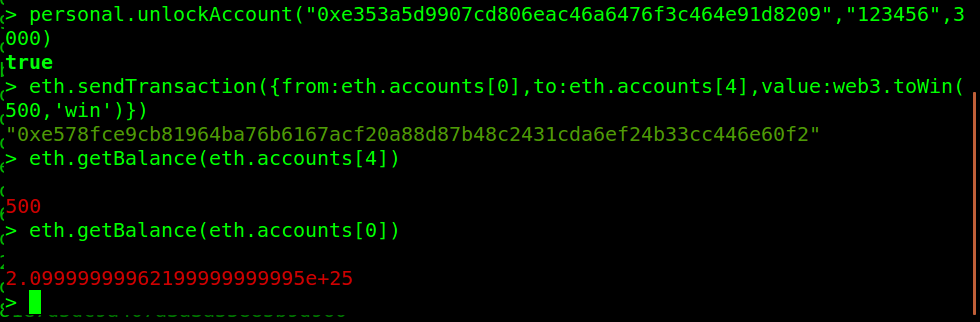
personal.unlockAccount("0xe353a5d9907cd806eac46a6476f3c464e91d8209","123456",3000)

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. 查看交易打包

****

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.

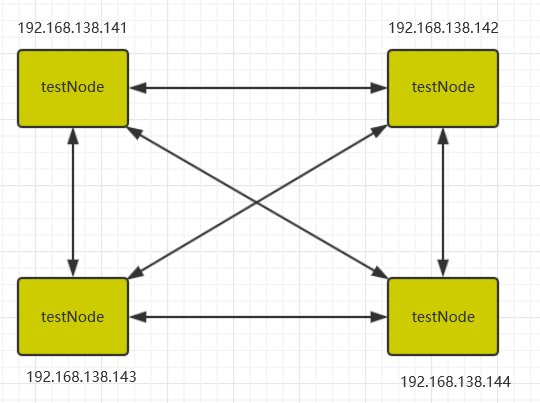
## 1.2多机器部署方法

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **节点名称** | **数据文件路径** | **节点端口号** | **rpc端口号** | **rpc地址** | **网络ID** |
| testNode | /root/go/testNode | 40000 | 50000 | 192.168.138.141 | 168 |
| testNode | /root/go/testNode | 40000 | 50000 | 192.168.138.142 | 168 |
| testNode | /root/go/testNode | 40000 | 50000 | 192.168.138.143 | 168 |
| testNode | /root/go/testNode | 40000 | 50000 | 192.168.138.144 | 168 |

### 1.2.1网络拓扑

节点配置如下:

拓扑关系如下:



### 1.2.2 环境配置

#### 1.2.2.1 为4个节点创建数据文件夹

在各节点上创建节点数据文件夹 mkdir /root/go/testNode

#### 1.2.2.2 交易账号准备

参见1.1.2.2章节.

#### 1.2.2.3 生成节点私钥

参见1.1.2.3章节.

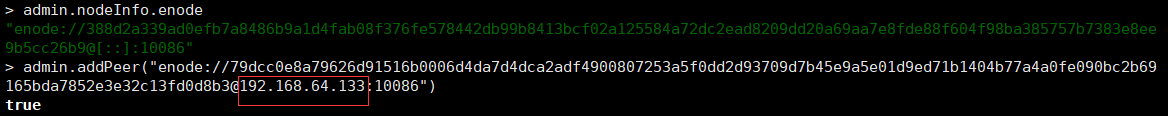
#### 1.2.2.4 IBFT创世块文件配置

参见1.1.2.4章节.

#### 1.2.2.5 静态节点列表生成

参见1.1.2.5章节.

**生成文件之后需要把文件中每个节点的enode地址中[::]替换成节点真实ip地址,例如:**

****

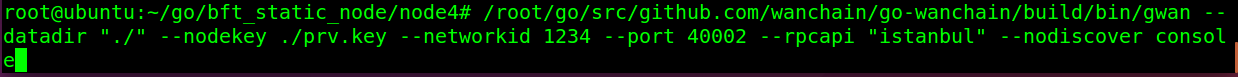
### 1.2.3 私链配置与运行

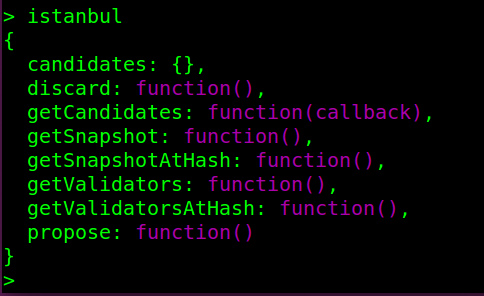
参见1.2.3章节.

## 1.3 istanbul web3.js扩展API使用示例

### 1.3.1 使能istanbul web3.js扩展

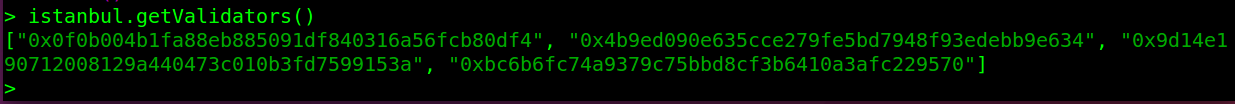
在启动gwan控制台的时候使用参数 –rpcapi “istanbul”.



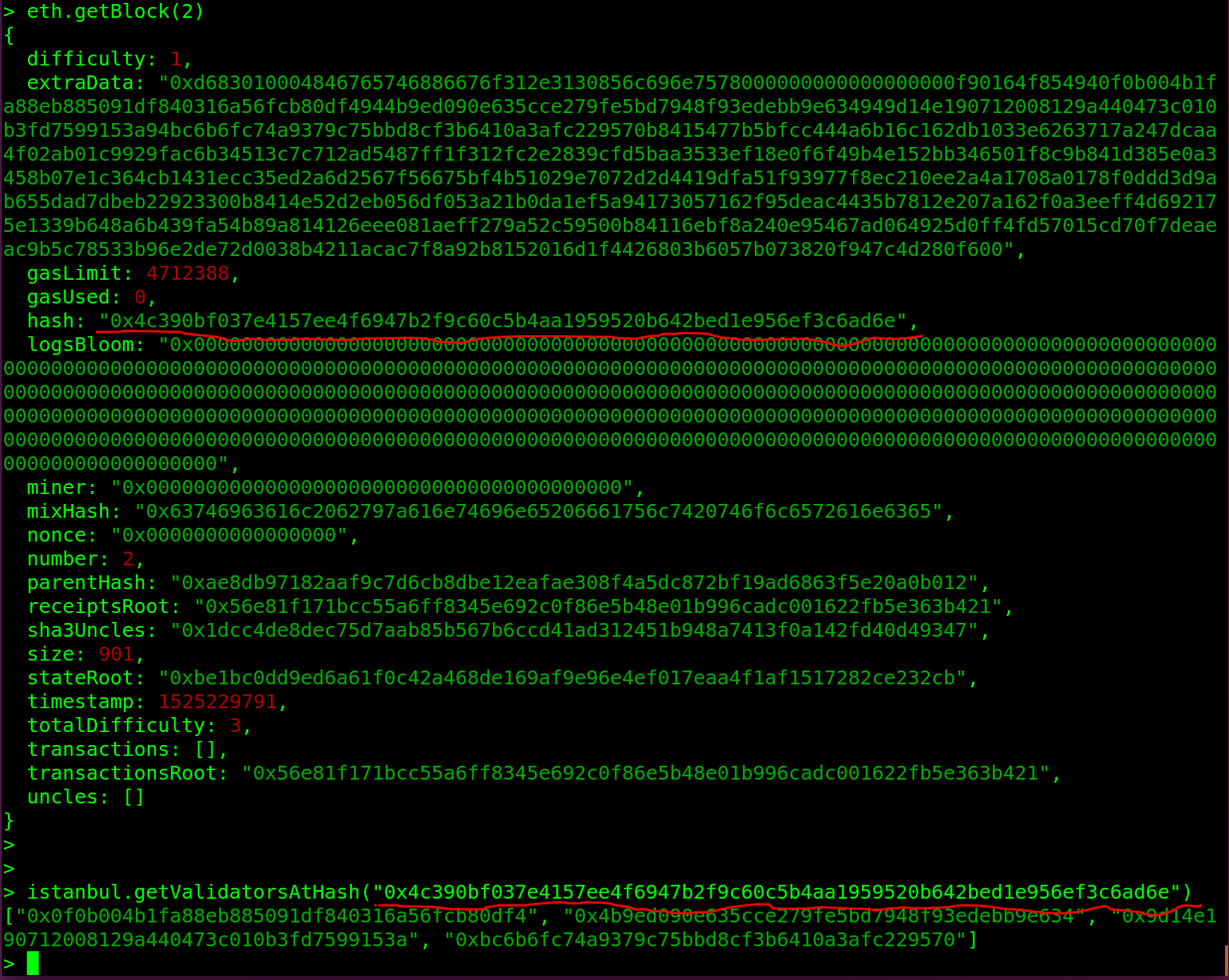


### 1.3.2 查看validators

Istanbul.getValidators():

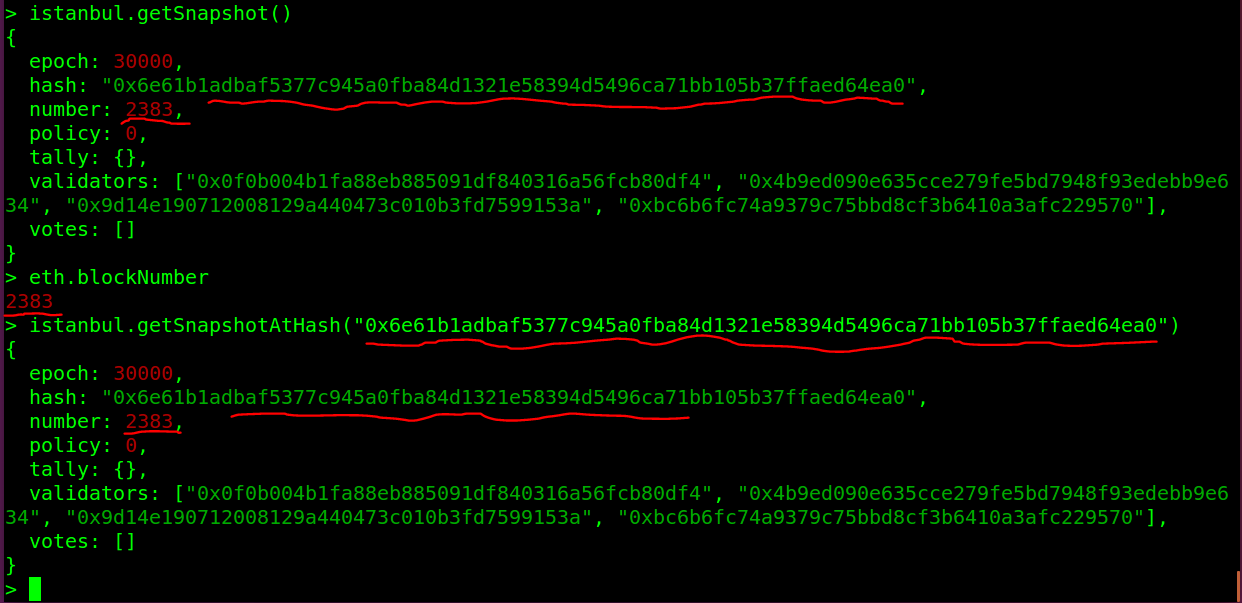


Istanbul.getValidatorsAtHash(<block hash>):



### 1.3.3 查看snapshot

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



### 1.3.4 动态添加/删除validators

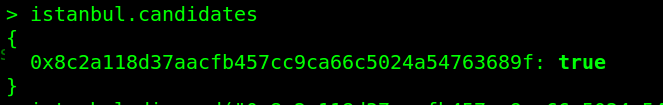


动态增加validators.

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>

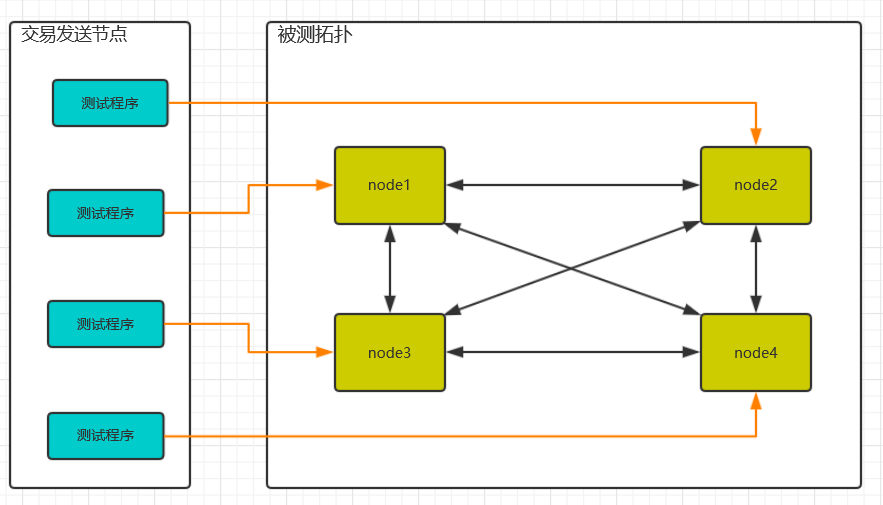
## 1.4性能测试结果

**说明:** 此性能测试采用单机4节点的部署方法, 从与之相连的独立机器上发送交易.

### 1.4.1 机器配置

|  |  |
| --- | --- |
| 操作系统 | Ubuntu 16.04.4 LTS |
| 内核 | Linux localhost 4.4.0-122-generic x86\_64 |
| 内存 | 16G |
| CPU | 8 \* Intel(R) Core(TM) i7 CPU 965 @ 3.20GHz |
| 硬盘 | N/A |

### 1.4.2 网络拓扑



### 1.4.3 测试结果

|  |  |  |
| --- | --- | --- |
| 测试目的 | 结果 | 说明 |
| 每分钟出块数 | 65个/分钟 | 各系统参数均未达到瓶颈 |
| 每分钟简单交易数量 (转账) | 13000笔交易/分钟 | 各系统参数均未达到瓶颈 |
| 每分钟复杂交易数 (执行职能合约) | 1100笔交易/分钟 | 各系统参数均未达到瓶颈 执行速率随合约复杂度增加而减小. |

在测试复杂交易中采用采用的智能合约如下:

pragma solidity ^0.4.0;

contract HelloWorldContract {

uint result;

function sayHi() constant returns (uint) {

for(uint i = 0; i < 1; i++) {

result = result +1;

}

return result;

}

}

# 备注

## Istanbul tool

Istanbul tool源代码链接:

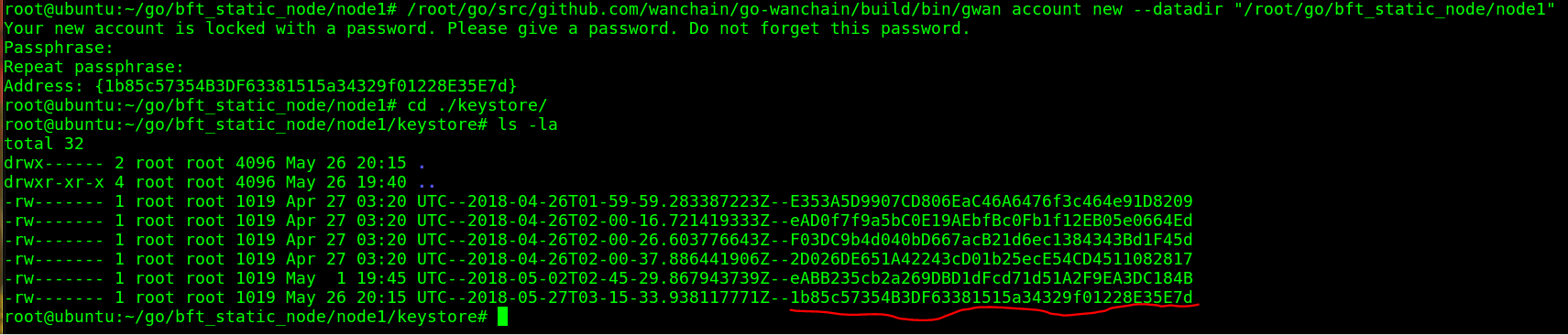
<https://github.com/getamis/istanbul-tools.git>

编译使用方法请参考此工具的readme.

## 账号创建

gwan account new --datadir "/root/go/bft\_static\_node/node1"

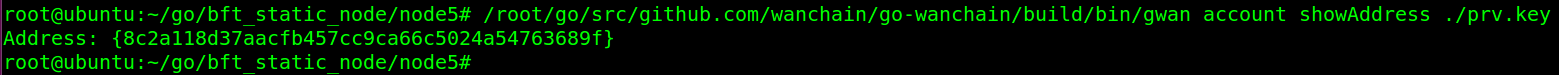
(创建好的账户位于节点node1的数据路径下的keystore文件夹.)



## IBFT共识节点列表配置

1. 获取4个节点nodekey对应的address.

gwan account showAddress ./prv.key



(showAddress使用方法参见备注-showAddress使用方法章节.)

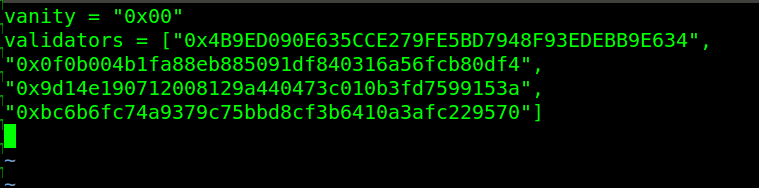
本实践中使用的节点nodekey对应的address如下:

|  |  |
| --- | --- |
| 节点名称 | 地址 |
| node1 | "0x4B9ED090E635CCE279FE5BD7948F93EDEBB9E634" |
| node2 | "0x0f0b004b1fa88eb885091df840316a56fcb80df4" |
| node3 | "0x9d14e190712008129a440473c010b3fd7599153a" |
| node4 | "0xbc6b6fc74a9379c75bbd8cf3b6410a3afc229570" |

1. 使用vim或其他编辑工具生成toml文件(文件名任意,路径任意, 例如: bft.toml)

Toml文件供后续步骤使用生成extraData.

Toml文件格式如下(validators中为步骤1生成的nodekey地址):

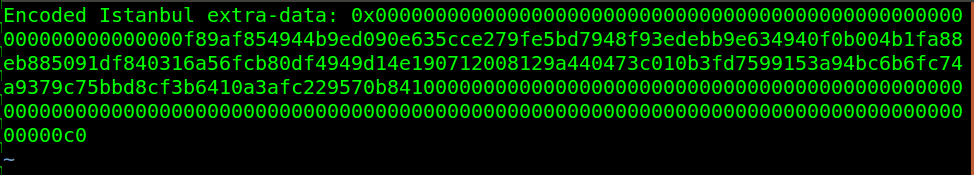


1. 通过istanbul\_tools生成用于写入创世块文件的 extraData.

(Istanbul\_tools使用方法参考”备注-Istanbul tool”章节)

/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:



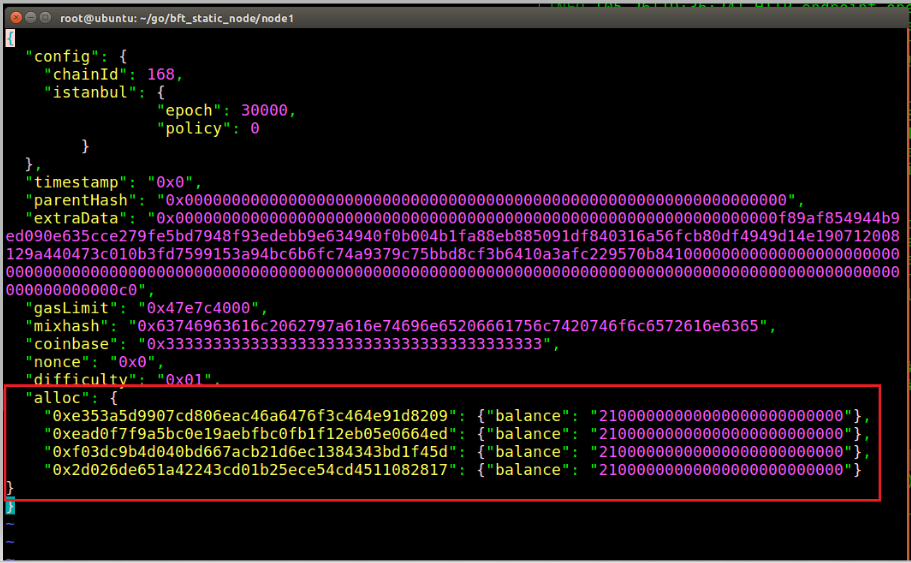
其中”Encoded Istanbul extra-data:”后面的内容为经过RLP编码的validators,需要拷贝到创世块文件genesis.json的extraData字段.

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

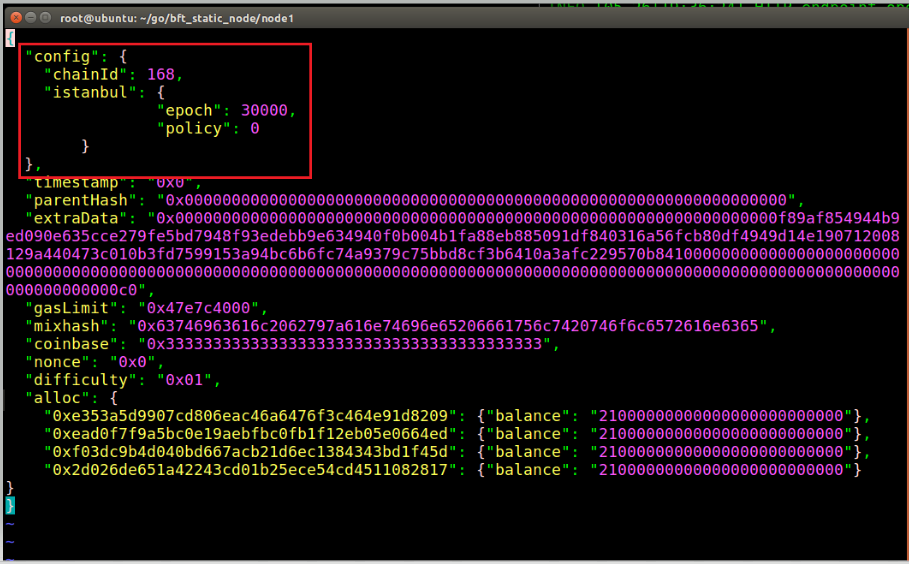
更新后的genesis.json内容如下:



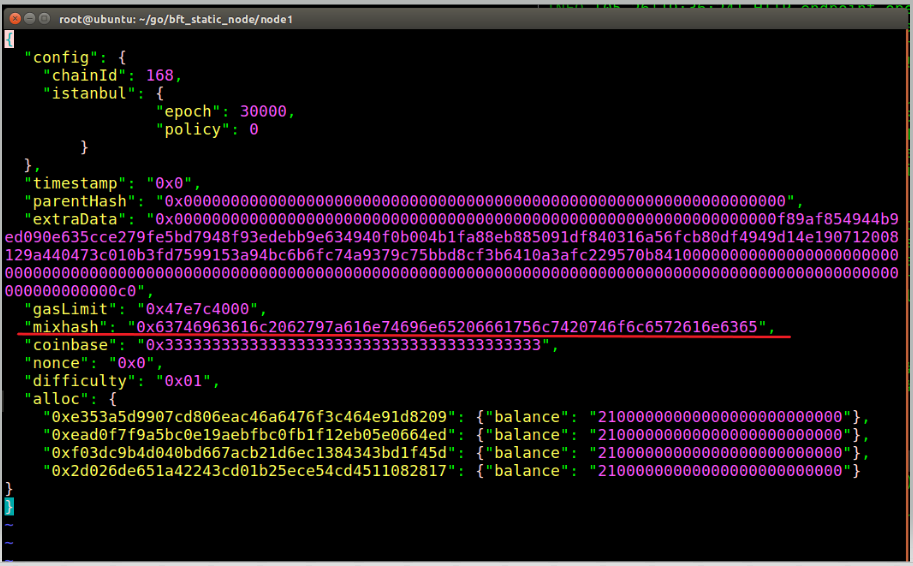
## IBFT交易账户配置



## IBFT属性配置



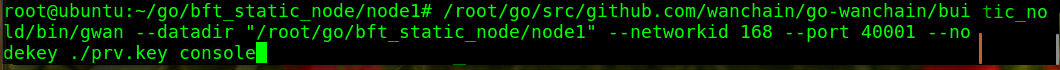
## IBFT mixhash配置



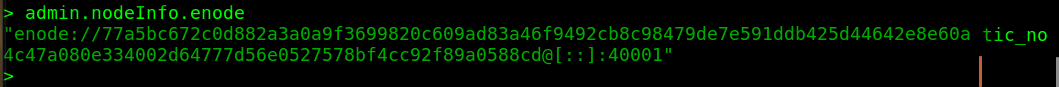
## 获得节点enode

* 1. 执行以下命令进入个节点命令行.

/root/go/src/github.com/wanchain/go-wanchain/build/bin/gwan --datadir "/root/go/bft\_static\_node/node1" --networkid 168 --port 40001 --nodekey ./prv.key console

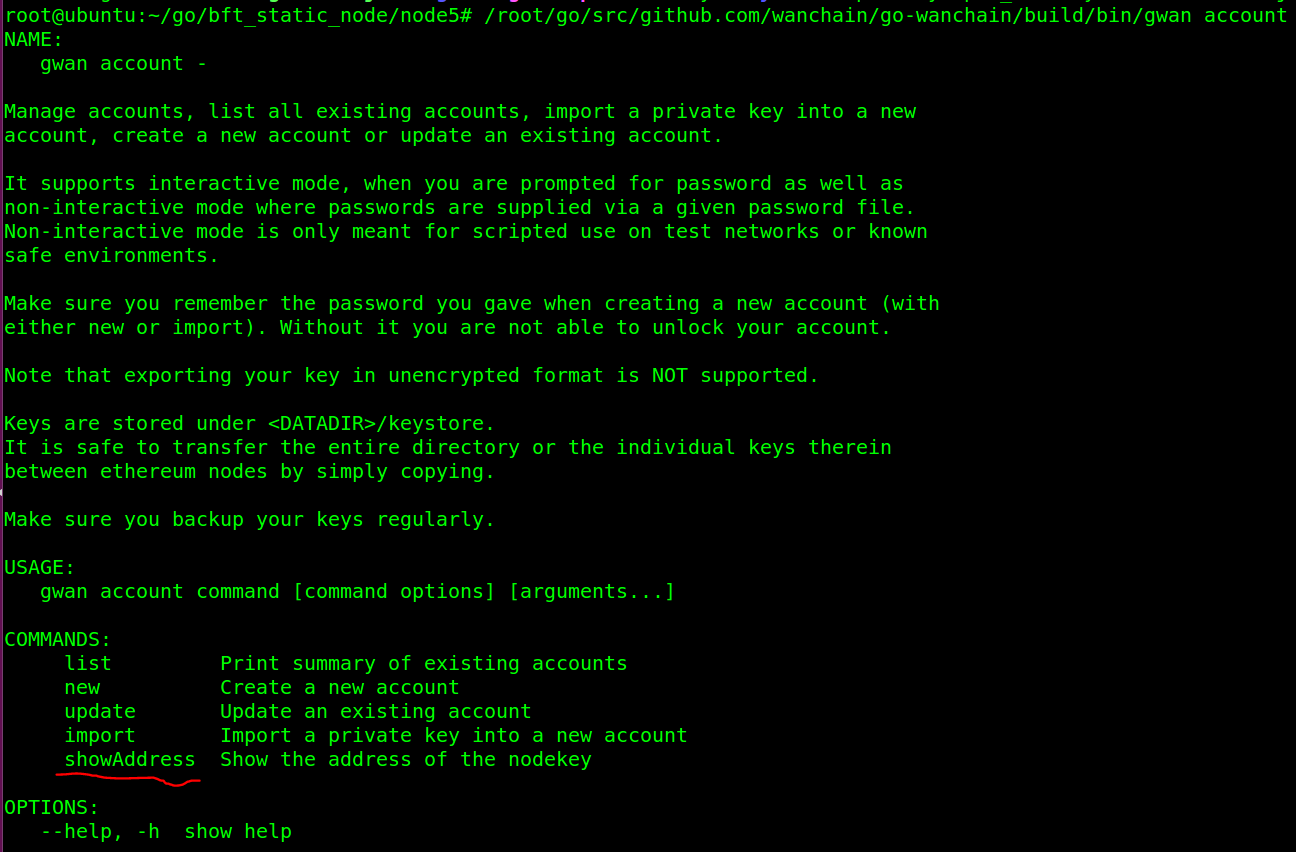


* 1. 在节点命令行控制台中执行命令admin.nodeInfo.enode查看节点enode.



## showAddress使用方法

显示account命令帮助信息:



显示showAddress命令帮助信息:



---- END ----

Todo:

* + - 1. 网络拓扑阶段形成一个表格.
      2. 准备工作部分的详细步骤放在备注中, 在准备工作一节中可以引用备注中的内容.
      3. 多机的例子中文件夹等名称需要一样,步骤不能按照单机的例子一样省略.
      4. 获得enode的方式,步骤调整.