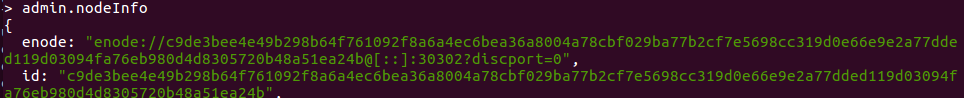
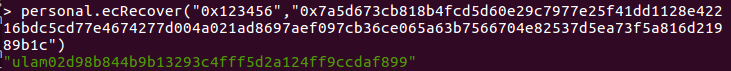
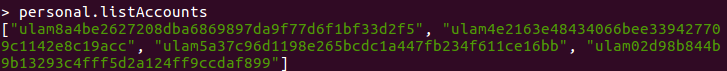
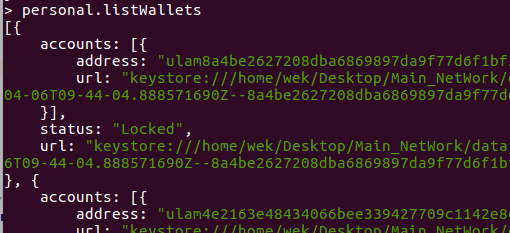
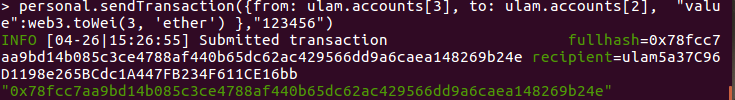
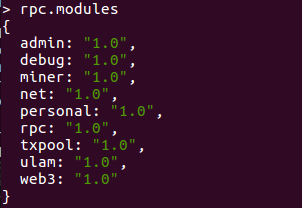
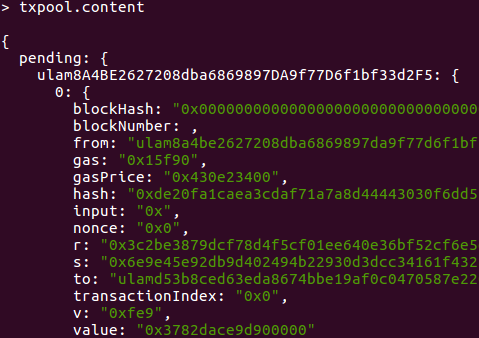
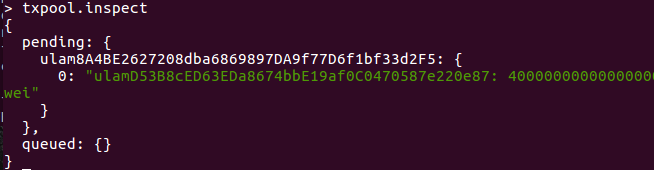
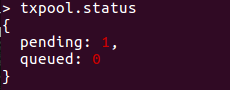
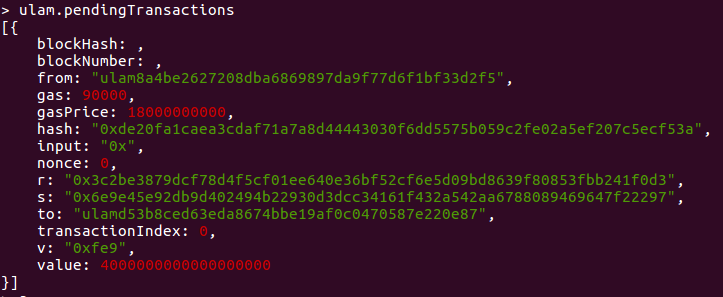
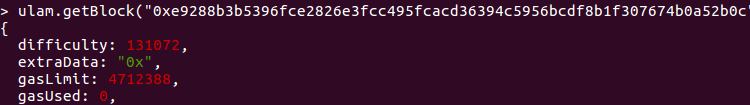
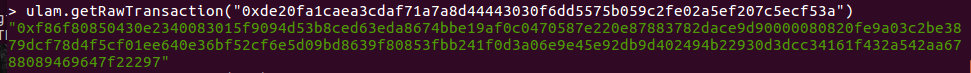
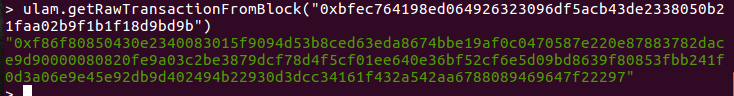
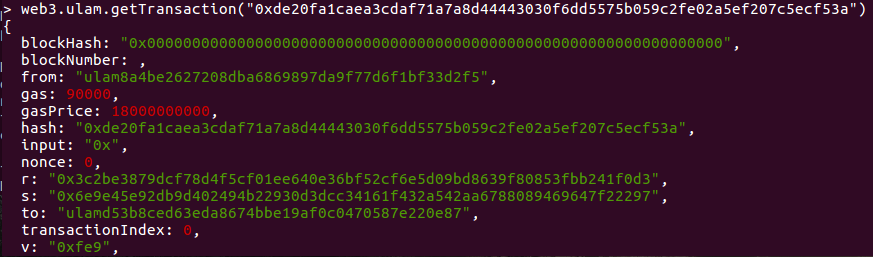
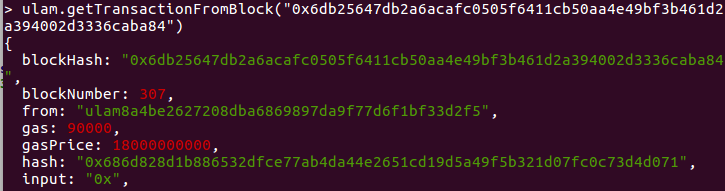
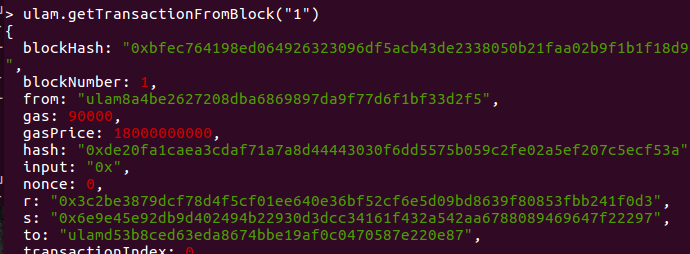
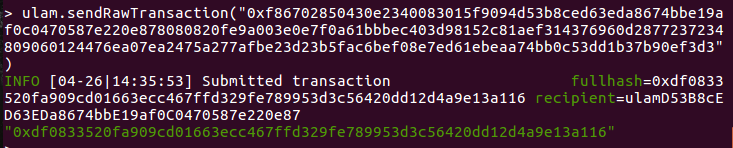
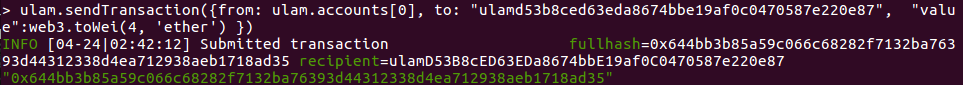
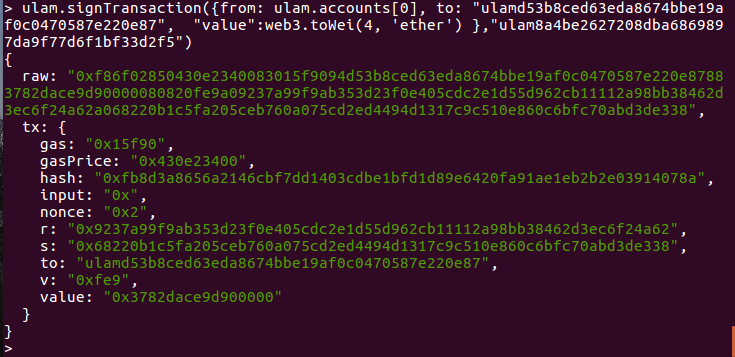
1. 配置环境
   1. Ubuntu系统
   2. ulam文件夹（两个文件，一个genesis.json，一个gulam可执行文件）
2. 操作步骤
   1. 在ulam文件夹下打开终端
   2. 执行命令：
      1. ./gulam –datadir data init genesis.json
      2. ./gulam –datadir data --rpc --rpcaddr IPaddress --rpcport 10011 --networkid 2019 --nodiscover --rpcapi "db,ulam,net,web3,admin" console
      3. admin.addPeer(MainNetEnode)
3. 操作命令

* admin
  1. peers ：获取已连接远程节点信息
     1. 参数：无参数
     2. 返回值：连接的远程节点旳网络信息。
     3. 用法
        1. 
  2. addPeer ：添加远程节点
     1. 参数：远程节点旳enode地址
     2. 返回值： 添加成功后会返回TRUE， 否则返回false
     3. 用法：
        1. 
  3. datadir ：获取链库数据目录
     1. 参数：无参数
     2. 返回值：获取geth节点旳数据存储路径
     3. 用法
        1. 
  4. nodeInfo ：获取节点信息
     1. 参数：无参数
     2. 返回值：当前运行的geth节点旳网络相关信息。
     3. 用法：
        1. 
  5. removePeer ：移除节点
     1. 参数：node id
     2. 返回值：ture
     3. 用法
        1. 
  6. sleep ：function github.com/ulam/go-ulam/console.(\*bridge).Sleep-fm()
  7. sleepBlocks ：function github.com/ulam/go-ulam/console.(\*bridge).SleepBlocks-fm()
  8. startRPC ：开启 HTTP 上的 RPC 服务
     1. 参数：所有的参数都是可选的
        1. host ：要监听的网络地址，默认值：localhost
        2. port：要监听的网络端口，默认值：8545
        3. cors：要使用的跨源资源共享头，默认值：""
        4. apis:  要透过该服务接口提供服务的API模块，默认值："eth,net,web3"
     2. 返回值：返回一个布尔值来表示HTTP RPC监听是否正确启动。
     3. 用法
        1. 
  9. startWS ：启动 WebSocket 上的 RPC 服务
     1. 参数：所有的参数都是可选的
        1. host ：要监听的网络地址，默认值：localhost
        2. port：要监听的网络端口，默认值：8545
        3. cors：要使用的跨源资源共享头，默认值：""
        4. apis:  要透过该服务接口提供服务的API模块，默认值："eth,net,web3"。
     2. 返回值：返回一个布尔值来表征webSocket上的RPC监听是否启动成功。
     3. 用法
        1. 
  10. stopRPC ：停止 HTTP 上的 RPC 服务
      1. 参数：无参数
      2. 返回值：一个布尔值，表示端节点是否成功关闭
      3. 用法
         1. 
  11. stopWS ：停止 WebSocket 上的 RPC 服务
      1. 参数：无参数。
      2. 返回值：表示是否成功关闭端结点的布尔值。
      3. 用法：
         1. 
* net
  1. listening ：如果客户端处于监听网络连接的状态，该调用返回true
     1. 参数：无参数。
     2. 返回值：节点处于网络监听状态时返回true， 否则返回false。
     3. 用法：
        1. 
  2. peerCount：返回当前客户端所连接的对端节点数量。
     1. 参数：无参数。
     2. 返回值：所连接其他节点旳数量。
     3. 用法
        1. 
  3. version ： 返回当前连接网络的ID。
     1. 参数：无参数
     2. 返回值：当前节点所处网络ID。
     3. 用法
        1. 
  4. getListening : function(callback)
  5. getPeerCount ：function(callback)
  6. getVersion : function(callback)
* personal
  1. ecRecover：从签名的数据中提取进行签名的账户地址
     1. 参数：
        1. data：要签名的数据。
        2. string：数据的签名。
     2. 返回值：为数据签名的账户地址。
     3. 用法
        1. 
  2. listAccounts : 列出所有账户
     1. 参数：无参数
     2. 返回值：密钥库中所有密钥对应的账户地址。
     3. 用法
        1. 
  3. getListAccounts ：function(callback)
  4. getListWallets：function(callback)
  5. listWallets:
     1. 参数：无参数
     2. 返回值：密钥库中所有密钥对应的账户状态及信息。
     3. 用法
        1. 
  6. importRawKey : 导入私钥
     1. 参数：
        1. Pricvatekey：私钥
        2. Password：密码
     2. 返回值：ulam地址
     3. 用法
        1. 
  7. lockAccount : 锁定账户，从内存中删除具有给定地址的私钥。该帐户不能再用于发送交易。
     1. 参数：账户地址
     2. 返回值：ture
     3. 用法
        1. 
  8. newAccount : 创建新账户
     1. 参数：password：密码
     2. 返回值：账户地址
     3. 用法
        1. 
  9. openWallet ：function github.com/ulam/go-ulam/console.(\*bridge).OpenWallet-fm(),
  10. sendTransaction : 发送交易
      1. 参数：
         1. Tx：交易
         2. Password：发送者密码
      2. 返回值：交易hash
      3. 用法
         1. 
  11. sign : 签名
      1. 参数：
         1. Massage：需要签名的消息
         2. Addr：账户地址
         3. Password：账户密码
      2. 返回值：签名后的数据
      3. 用法
         1. 
  12. unlockAccount : 解锁账户
      1. 参数：
         1. Address：账户地址
         2. Password：密码
      2. 返回值：一个bool值，代表是否解锁成功。
      3. 用法
         1. 
* rpc
  1. getModules ： function(callback)
  2. Modules
     1. 用法
        1. 
* Txpool
  1. content : 获取交易池中交易详情
     1. 参数：无参数
     2. 返回值：交易池中交易详情
     3. 用法
        1. 
  2. inspect ：获取池中交易概述
     1. 参数：无参数
     2. 返回值：交易池中的交易概述
     3. 用法
        1. 
  3. status ：获取交易池状态
     1. 参数，无参数
     2. 返回值：交易池状态
     3. 用法
        1. 
  4. getContent ：function(callback)
  5. getInspect ：function(callback)
  6. getStatus ：function(callback)
* ulam
  1. accounts：获取当前节点钱包的所有账户
     1. 参数：无参数
     2. 返回值：当前节点持有的帐户列表
     3. 用法：
        1. 
  2. blockNumber：返回当前块高度
     1. 参数：无参数
     2. 返回值：返回当前链的区块高度
     3. 用法
        1. 
  3. defaultBlock：默认区块
     1. 参数：无参数
     2. 返回值：默认返回值为“latest”
     3. 用法
        1. 
  4. pendingTransactions：返回正在挂起的交易列表
     1. 参数：无参数
     2. 返回值：正在挂起的交易列表
     3. 用法
        1. 
  5. getAccountInfo ： 获取账户信息
     1. 参数：addr：账户地址
     2. 返回值：账户信息
     3. 用法：
        1. 
  6. getAccounts ： function(callback)
  7. getBalance：获取余额
     1. 参数：addr：地址
     2. 返回值：账户的余额
     3. 用法
        1. 
  8. getBlock：获取区块
     1. 参数：
        1. Blockhash或者blockNumber
        2. Boolean ：可选，默认值为false。当设置为true时,返回块中将包括所有交易详情，否则仅返回交易哈希。
     2. 返回值：区块信息
     3. 用法
        1. 
  9. getBlockNumber：function(callback)
  10. getBlockTransactionCount
      1. 参数：
         1. blocknumber或者blockhash
         2. "genesis"、"latest" 或 "pending" 来指定块。
      2. 返回值：指定区块的交易数
      3. 用法
         1. 
  11. getPendingTransactions：function(callback)
  12. getRawTransaction
      1. 参数：交易hash
      2. 返回值：交易的签名数据
      3. 用法
         1. 
  13. getRawTransactionFromBlock
      1. 参数：区块hash
      2. 返回值：交易签名
      3. 用法
         1. 
  14. getTransaction：获取交易信息
      1. 参数：交易hash
      2. 返回值：交易
      3. 用法
         1. 
  15. getTransactionCount：获取交易数
      1. 参数：addr : 账户地址
      2. 返回值：指定地址的交易数
      3. 用法
         1. 
  16. getTransactionFromBlock
      1. 参数：blockhash或者blocknumber
      2. 返回值：指定区块的交易
      3. 用法
         1. 
         2. 
  17. getTransactionReceipt：获取交易凭证
      1. 参数：交易hash
      2. 返回值：
      3. 用法
         1. 
  18. resend：重新发送交易
  19. sendRawTransaction：发送一个已签名的交易
      1. 参数：签名后的交易
      2. 返回值：交易hash
      3. 用法
         1. 
  20. sendTransaction：发送交易
      1. 参数：tx ：交易
      2. 返回值：交易hash
      3. 用法
         1. 
  21. sign：对data进行签名，用来签名的账户地址需要首先解锁
      1. 参数
         1. Addr：签名账户
         2. Massage：待签名的数据
      2. 返回值：数据的签名
      3. 用法
         1. 
  22. signTransaction：对交易进行签名，用来签名的账户地址需要首先解锁
      1. 参数：
         1. 待签名的tx :交易
         2. 签名的账户地址
      2. 返回值：签名信息及交易
      3. 用法
         1. 
* web3
  1. bzz：和去中心化文件存储系统Swarm交互。
     1. hive
     2. info
     3. blockNetworkRead
     4. download：从swarm网络下载文件和目录到缓冲buffer中，或者磁盘中
     5. get
     6. getHive
     7. getInfo
     8. modify
     9. put
     10. retrieve
     11. store
     12. swapEnabled
     13. syncEnabled
     14. upload：将文件夹或裸数据上传到swarm网络。
  2. currentProvider
     1. newAccount
     2. openWallet
     3. send
     4. sendAsync
     5. sign
     6. unlockAccount
  3. db
     1. getHex： 从本地leveldb数据库中读取指定键的16进制格式字符串.
     2. getString：从本地leveldb数据库获取字符串
     3. putHex：本地leveldb数据库中以二进制存储方式存入 16进制字符串
     4. putString：在本地leveldb数据库中存储字符串.
  4. isIBAN
  5. providers：返回当前有效的通信服务提供器。
     1. HttpProvider
     2. IpcProvider
  6. Settings
     1. defaultAccount：默认账户
     2. defaultBlock：默认区块
  7. shh：用来使用whisper协议进行广播发送。
     1. addPrivateKey：根据给定的私钥生成密钥对，并在保存后返回其ID。
     2. addSymKey：保存对称密钥并返回其ID。
     3. deleteKeyPair：删除指定的键。
     4. deleteSymKey：删除对称密钥。
     5. generateSymKeyFromPassword：使用指定密码生成对称密钥，保存并返回ID。
     6. getPrivateKey：返回指定ID密钥对的私钥。
     7. getPublicKey：返回指定ID密钥对中的公钥。
     8. getSymKey：返回指定ID关联的对称公钥。
     9. hasKeyPair：检查whisper节点的私钥是否匹配给定的ID。
     10. hasSymKey：检查指定的ID是否存有对称密钥。
     11. info
     12. markTrustedPeer：将指定的节点标记为可信，从而允许其发送历史（过期）消息
     13. newKeyPair：生成用于消息加密和解密的密钥对。
     14. newMessageFilter：在节点内创建一个新的过滤器，可用于轮询匹配条件集的新消息。
     15. newSymKey：生成随机对称密钥，保存后返回ID，该对称秘钥用于消息加密和解密。
     16. post：向ulam网络发送whisper消息
     17. setMaxMessageSize：设置节点允许的消息大小上限
     18. setMinPoW：设置节点允许的最小POW值
     19. version
  8. BigNumber
  9. createBatch：批量请求，可以允许将多个请求放入队列，并一次执行。
  10. fromAscii：将ascii转为16进制
  11. fromDecimal：将numeric转为16进制
  12. fromICAP
  13. fromUtf8：将utf8转为16进制
  14. fromWei：返回wei转换为给定货币的值
  15. isAddress：如果该值是已识别的地址格式之一，则返回true
  16. isChecksumAddress：如果该值符合EIP55而且是有效的地址，则返回true
  17. isConnected：可以用来检查到节点的连接是否存在（connection to node exist）。
  18. padLeft：在指定字符串左侧补0到指定长度。
  19. padRight：在指定字符串右侧补0到指定长度。
  20. reset：用来重置web3的状态。重置除了manager以外的其它所有东西。卸载filter，停止状态轮询。
  21. setProvider：设置网络通信服务提供对象。
  22. sha3：计算给定字符串的sha3哈希值
  23. toAscii：将16进制字符转为Ascii进制
  24. toBigNumber：将给定的数字或十六进制字符串转为BigNumber
  25. toChecksumAddress：将给定的大写或小写地址转换为校验和地址
  26. toDecimal：将16进制转为numeric
  27. toHex：将16进制转为ascii。
  28. toUtf8：将16进制转为utf-8
  29. toWei：将指定单位的币额换算成wei