05/04/2022

OTEx end-to-end geth local Blockchain Setup

1. Setup geth 1.9.1 with all tools:

1.1 Visit https://geth.ethereum.org/downloads/ and download the tar file "Geth and Tools 1.9.1" for 64-bit.

1.2 Extract the tar file using the following command:
sudo tar -xvf <filename>
eg: sudo tar -xvf geth-alltools-linux-amd64-1.9.1-b7b2f60f.tar.gz

1.3 Step into extracted folder
cd geth-alltools-linux-amd64-1.9.1-b7b2f60f

1.4 Make the geth and puppeth files executable with the below command:
sudo chmod +x geth sudo chmod +x puppeth

1.5 Copy file to the user bin:
sudo cp geth /usr/local/bin/ sudo cp puppeth /usr/local/bin/

1.6 Check the geth version:
geth version

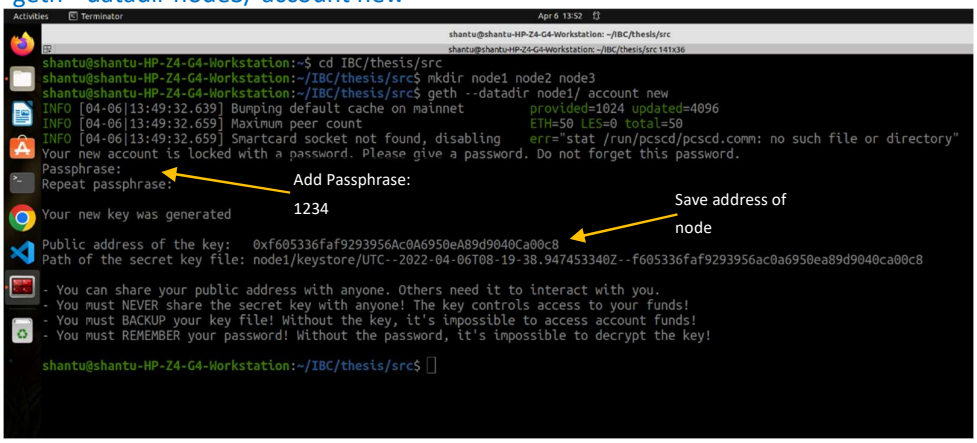
2. Setting up nodes and accounts:

2.1 make directory and get into:
mkdir IBC cd IBC

2.2 make src, dst and aux directories in IBC:
mkdir src dst aux

2.3 make node directories inside src:
cd src
mkdir node1 node2 node3
node1: alice, node2: srcGateway, node3: charlie

2.4 Setup accounts in each of the nodes and give each of them a password (preferably 1234 according to the script):
geth --datadir node1/ account new
geth --datadir node2/ account new
geth --datadir node3/ account new



3. Configuring genesis file using puppeth:

3.1 Launch puppeth by using 'puppeth' in the terminal:

3.2 Enter the name of genesis file

```
| This tool lets you create a new Ethereum network down to  |
| the genesis block, bootnodes, miners and ethstats servers |
| without the hassle that it would normally entail.         |
|                                                           |
| Puppeth uses SSH to dial in to remote servers, and builds |
| its network components out of Docker containers using the |
| docker-compose toolset.                                   |
+-----------------------------------------------------------+

Please specify a network name to administer (no spaces, hyphens or capital letters please)
> gen_src
```

3.3 Choose 2 to 'Configure new genesis' and then 1 to 'Create new genesis from scratch'

```
INFO [04-06|13:55:21.355] Administering Ethereum network           name=gen_src
INFO [04-06|13:55:21.355] No remote machines to gather stats from

What would you like to do? (default = stats)
 1. Show network stats
 2. Configure new genesis
 3. Track new remote server
 4. Deploy network components
> 2

What would you like to do? (default = create)
 1. Create new genesis from scratch
 2. Import already existing genesis
> 1
```

3.4 Choose clique option for a POA blockchain

```
Which consensus engine to use? (default = clique)
 1. Ethash - proof-of-work
 2. Clique - proof-of-authority
> 2
```

3.5 Enter time per block (preferably small, 5)

3.6 Enter address of the nodes for whom we want to make a miner. For us we need to make all the nodes as miners.

3.7 Accounts needed to be pre-funded

```
Which accounts are allowed to seal? (mandatory at least one)
> 0xf605336faf9293956Ac0A6950eA89d9040Ca00c8
> 0xD7faB1684ecEf0156c8794b6626Fb893E48b9d10
> 0x0164299f321D2008867721aD312998271eF99277
> 0x

Which accounts should be pre-funded? (advisable at least one)
> 0xf605336faf9293956Ac0A6950eA89d9040Ca00c8
> 0xD7faB1684ecEf0156c8794b6626Fb893E48b9d10
> 0x0164299f321D2008867721aD312998271eF99277
> 0x
```

3.8 Pre-fund the accounts by typing 'yes'

3.9 Export genesis file to current directory

4. Initialising and Running the nodes

4.1 Initialise geth in each of the nodes

geth --datadir node1/ init src_gen.json

geth --datadir node2/ init src_gen.json

geth --datadir node3/ init src_gen.json

4.2 Run the nodes with mining switched on

geth --datadir <node dir> --syncmode 'full' --port <port #> --rpc --rpcaddr 'localhost' --rpcport <rpc port #> --rpcapi 'personal,debug,eth,net,web3,txpool,miner' --networkid <networkid> --unlock <node address> --password "<path to password text file>" --allow-insecure-unlock --nodiscover

<node dir> path to respective nodes (node1/, node2/, ...)
<port #> different for all 3 nodes (30511, 30512, 30513)
<rpc port #> different for all 3 nodes (8701, 8702, 8703)
<node address> can be found in keystore and noted before
<path to password txt file>, txt file contains the password used while creating the accounts (eg: node1/password.txt)

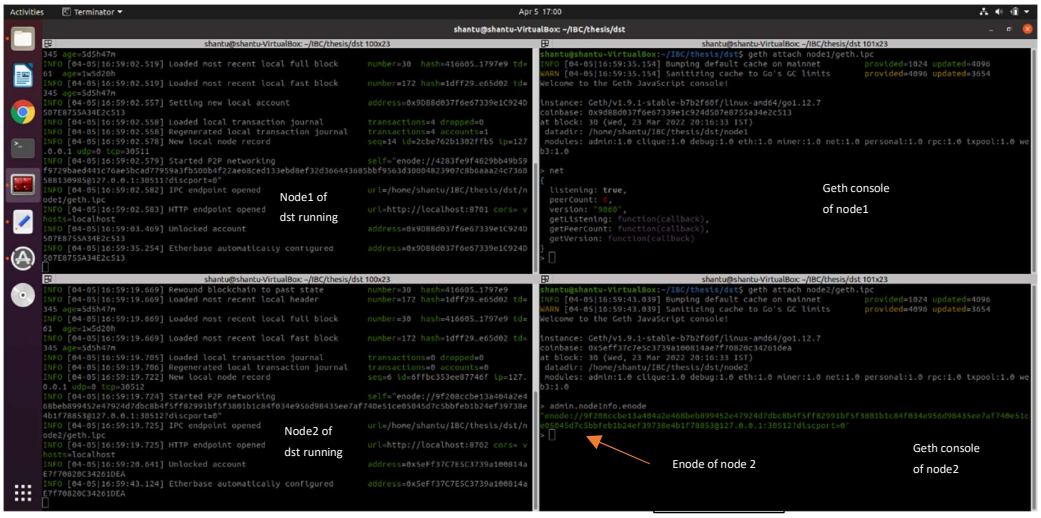
geth --datadir node1 --syncmode 'full' --port 30511 --rpc --rpcaddr 'localhost' -- rpcport 8701 --rpcapi 'personal,debug,eth,net,web3,txpool,miner' --networkid 9060 --unlock 0x9D88d037f6e67339e1C924D507E8755A34E2c513 --password "/home/shantu/IBC/dst/node1 password.txt" --allow-insecure-unlock --nodiscover

5. Adding Peers:

5.1 Attach a geth console to each of running nodes

geth attach node1/geth.ipc

For each node have geth console in different terminals



5.2 Check number of peers of node using 'net'

5.3 Making each node as peers

5.3.1 get enode of node1 using the command
admin.nodeInfo.enode

5.3.2 add enode of node1 in node2 using the command
admin.addPeer("<enode of node1>")

5.3.3 add enode of node2 in node3 like in the above steps

5.3.4 Check number of peers of the nodes using 'net'

5.4 Make the default as the coinbase for all nodes using:
eth.default = eth.coinbase

6. Repeat the steps 1-5 for dst and aux. Only two nodes are enough for dst and aux.

7. Add the 'htlc_abi.txt' and 'token_abi.txt' and the 'ibc_v1.2.py' files to the IBC folder. Make the changes to 'ibc_v1.2.py' marked in comments i.e., changing the address of alice and srcGateway.

```
with open('./token_abi.txt') as fp:
    token_abi=json.loads(fp.read())
token_instance=
src2.eth.contract(address=Web3.toChecksumAddress("0x2547CA6e8D265075FBD33dF863bC205f83a9B8A4"
abi=token_abi)#,ContractFactoryClass=ConciseContract) #address of srcGateway (node2)

with open('./htlc_abi.txt') as fp:
    htlc_abi=json.loads(fp.read())
htlc_instance=
src1.eth.contract(address=Web3.toChecksumAddress("0x5EeD093aE3EB2136366Ac1E80C1b05fAC120e883"
abi=htlc_abi)#,ContractFactoryClass=ConciseContract) #address of Alice (node1) here
print("Lock Tokens Smart Contract instance is created\n\n")


src1.geth.personal.unlock_account(src1.eth.accounts[0], "1234") #password of srcGateway
(node1)
src2.geth.personal.unlock_account(src2.eth.accounts[0], "1234") #password of srcGateway
(node2)

print("---Alice Approves tokens to be issued----\n\n")
approve_tx_hash =
token_instance.functions.approve('0x5EeD093aE3EB2136366Ac1E80C1b05fAC120e883',
100).transact({'from': src2.eth.accounts[0]}) #address of Alice (node1) here
print("Approve Tokens Tx_Hash:", approve_tx_hash.hex())
print()
```