OSC Nuco Binary Instructions

Join Nuco network

1. Uncompress the binary

```
tar -xvf nucog1.3.4.1.tar.gz
```

2. In nucog1.3.4.1 folder, export the environment

```
source ./setenv.sh
```

3. Setup a coinbase account:

```
./nucog account new
```

4. Run Nuco kernel

```
./nuco.sh
```

5. While your kernel is running, open another terminal and start the admin console

```
./admin.sh
```

6. In admin console, check if your node is connected

```
admin.peers
```

If it returns empty array, wait for 10 seconds and try it again, if there are node info in the array, you are successfully connected.

7. You can check our dashboard at http://monitor.nuco.io:7000/#/ for up to date block info

Stand alone node

1. Make sure your database is clean, in home directory:

```
rm -rf .nuco
```

2. In conf/nuco.cfg file, remove the enode information inside the nodes array, change the sealing value and the maxpeer value

```
nodes = [ ]
sealing = true
maxpeers = 0
maxpendingpeers = 0
```

3. In nucog1.3.4.1 folder, export the environment

```
source ./setevn.sh
```

4. Setup a coinbase account:

```
./nucog account new
```

5. Run Nuco kernel

```
./nuco.sh
```

6. While your kernel is running, open another terminal and start the admin console

```
./admin.sh
```

7. Check if you kernel is running alone

```
admin.peers
```

If an empty array is returned, you are running a stand alone network

Setup your private network

1. Make sure your database is clean, in home directory

```
rm -rf .nuco
```

2. In conf/nuco.cfg file, remove the enode information inside the nodes array, change the sealing value and the net ID value to any number other than 248

```
nodes = [ ]
sealing = true
netId = number
```

3. In nucog1.3.4.1 folder, export the environment

```
source ./setevn.sh
```

4. Setup a coinbase account:

```
./nucog account new
```

5. Run Nuco kernel

```
./nuco.sh
```

6. While your kernel is running, open another terminal and start the admin console

```
./admin.sh
```

7. Get at least one node info in your network

```
admin.nodeInfo
```

8. Copy this information in this format:

type://key@ip[:port]

e.g

"enode://89c529ac973eed040de7fff833f384d75e200fdf8a59afe7178d3b4940f973ae0e608b7ac50afd4c06a77cb0b589722dd6c5a182363456778a@199.53.177.75:30305"

9. In peers' admin console

```
admin.addPeer(nodeinfo)
```

10. Check if the nodes are connected:

```
admin.peers
```

If the console returns the node information you just added, you are successfully connected.

Deploy Your First Contract (token contract)

Prerequisites

- Make sure you have more than one account to transfer token to. To create a new account:

```
personal.newAccount()
```

- Make sure to let the kernel know which account to operate:

```
web3.eth.defaultAccount = web3.eth.coinbase
```

-Make sure your account is unlocked:

```
personal.unlockAccount(web3.eth.accounts[0])
```

- The time out for unlocked account is 3 minutes, to keep it unlocked for longer period:

```
personal.unlockAccount(web3.eth.accounts[0],'password','time')
```

Deploy Contract

1. Compile your contract. You need to remove all the line breaks and paragraph breaks of your contract to fit in a string variable. You can use some online tools to achieve it.

```
var source = ‘contract MyToken { mapping (address => uint256) public balanceOf; function MyToken( uint256 initialSupply ) { balanceOf[msg.sender] = 60000; function transfer(address _to, uint256 _value) { if (balanceOf[msg.sender] < _value) throw; if (balanceOf[_to] + _value < balanceOf[_to]) throw; balanceOf[msg.sender] -= _value; balanceOf[_to] += _value; } }}’
```

2. Prepare for deployment

```
var tokenCompiled = web3.eth.compile.solidity(source);
```

3. Construct your contract

```
var tokenContract = web3.eth.contract(tokenCompiled.myToken.info.abiDefinition);
```

4. Deploy your contract

```
var token = tokenContract.new(
{
from: web3.eth.defaultAccount,
data: tokenCompiled.myToken.code,
gas: 4700000
},
function(e, contract){
    if(!e) {
        if(!contract.address) {
        console.log("Contract transaction send: TransactionHash: " + contract.transactionHash + " waiting to be mined...");
        } else {
    console.log("Contract mined! Address: " + contract.address);
    console.log(contract);
        }
```

```
    }
});
```

Make sure you gas is equal or greater than 4700000

5. After deploying the contract, you will see a transaction hash return right away and the contract address will appear once the contract is successfully deployed.
6. Check initial balance

```
balanceOf(web3.eth.accounts[0])
```

7. Transfer token from default account to another account:

```
token.transfer(web3.eth.account[1], 10)
```

8. Check account 1's balance

```
balaceOf(web3.eth.accounts[1])
```

Transfer token within the network

1. Obtain ABI of the smart contract. Use solidity online compiler to get the ABI

```
var abi = “[{"constant":true,"inputs":
[{"name":"","type":"address"}],"name":"balanceOf","outputs":[{"name":"","typ
e":"uint256"}],"payable":false,"type":"function"},
{"constant":false,"inputs":[{"name":"_to","type":"address"},{"name":"_value","t
ype":"uint256"}],"name":"transfer","outputs":[],"payable":false,"type":"func
tion"},{"inputs":[{"name":"initialSupply","type":"uint256"}],"payable":false,"t
ype":"constructor"}]”
```

2. Deploy contract using the same contract address

```
var token = web3.eth.contract(abit).at(contract address)
```

3. Transfer token to another node

```
token.transfer(account address, amount)
```

Useful commands

Check your account:

```
eth.accounts
```

Check block number:

```
web3.eth.blockNumber
```

Get block information:

```
web3.eth.getBlock(block number)
```

Get transaction information:

```
web3.eth.getTransaction(transactionHash)
```

Check connected peers:

```
admin.peers
```