Solidity Assignment Phase-3

(Deploying Smart contract to geth using truffle)

Submitted by : Sumeet Pol

**Initialize private blockchain using geth:**

|  |
| --- |
| ***Note:***  *Skip the account creation step if you are using datadir1 provided with assignment* |

**Create folder for datadir**

$ mkdir datadir1

Create 3 accounts

$ geth --datadir datadir1/ account new

Provide password : password

$ geth --datadir datadir1/ account new

Provide password : password

$ geth --datadir datadir1/ account new

Provide password : password

|  |
| --- |
| ***Note:***  *Save the address generated using above commands, because it will be needed to unlock accounts to perform transactions* |

Initalize private blockchain

$ geth --datadir datadir1/ init genesis.json

Start geth console and unlock accounts for transactions

|  |
| --- |
| ***Note:***  *Accounts used in command are present in datadir1 provided with assignment*  *If you have created new accounts in new datadir then please provide them in command after --unlock flag eg.”address1”,”address2”,”address3”*  *Also use the same password which are set while creating accounts using account new command.* |

$ geth --datadir ./datadir1 --unlock "0xcca930d6213b3eb62707d918a9858c1f08de0761","0x4409f245bdcf62e40c3f5475803fa460d1173ad5","0x3cb0c18f3dec897c0eb8174c44017b0c698f6cb9" --rpc --rpcaddr localhost --rpcport 8545 console --rpcapi web3,eth,personal,miner,net,txpool

Start miner in geth console

miner.start(1)

|  |
| --- |
| ***Note:***  *Please terminal open where miner is running.*  *Open new Terminal for truffle under KYC-SC folder* |

Truffle commands for KYC deployment and interacting with contract

**Compile smart contarct**

$ truffle compile

**Migrate contract to our geth private network**

$ truffle migrate --network development

|  |
| --- |
| ***Note:***  *Assuming miner is running in other terminal. For above command*  *Note down the address where contract is deployed.*  *Deployed address will be needed to interact with contract using truffle console.* |

Truffle console interaction with KYC contract

**Truffle Console start**

$ truffle console --network development

**Create instance of kyc contract.**

|  |
| --- |
| ***Note:***  *If using datadir provided with assignment use below command directly*  *Else if you have setup new datadir then use the new deployed address from truffle migrate output.* |

truffle(development)> let kyc = await KYC.at("0xB17eB93869Ef1A710AEA0374F9D9fE6ADB5dd0Fe")

**Saving all accounts from private blockchain to accounts variable to further use**

truffle(development)> var accounts;

truffle(development)> web3.eth.getAccounts(function(err,res) { accounts = res; });

ADMIN INTERACTION

**Create Bank1**

truffle(development)> kyc.addBank("bank1","bid01",accounts[1],true)

**Create Bank2**

truffle(development)> kyc.addBank("bank2","bid02",accounts[2],true)

**Get the Bank name using bank.ethaddress**

truffle(development)> kyc.getBankName(accounts[1])

truffle(development)> kyc.getBankName(accounts[2])

**See the bank created using above command**

truffle(development)> kyc.allBanks(0)

**Blocking bank1 from submitting kyc request**

truffle(development)> kyc.blockBankFormSubmittingKyc(accounts[1])

**See the allowedKyc status is equal to false**

truffle(development)> kyc.allBanks(0)

**Blocking bank1 from submitting kyc request**

truffle(development)> kyc.unblockBankFormSubmittingKyc(accounts[1])

**See the allowedKyc status is equal to true**

truffle(development)> kyc.allBanks(0)

BANK INTERACTIONS:

|  |
| --- |
| ***Note:***  *Hence using from:accounts[1] which contains banks address* |

**Add Customer**

truffle(development)> kyc.addCustomer("sumeet","eed38600fe7dbd179cbaa86e00ab362a707b88d121120bb45e0891e0583e10f1",{from:accounts[1]})

**To See if customer is added**

truffle(development)> kyc.allCustomers(0)

**Update Customer information**

truffle(development)> kyc.modifyCustomer("sumeet","eed38600fe7dbd179cbaa86e00ab362a707b88d12112",{from:accounts[1]})

**To See if data is updated**

truffle(development)> kyc.allCustomers(0)

**Add KYC Request**

truffle(development)> kyc.addRequest("sumeet","0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C",{from:accounts[1]})

**To See if request is added**

truffle(development)> kyc.allRequests(0)

|  |
| --- |
| ***Note :***  *Current valid kyc threshold is* ***5*** *,*  *Hence when upvotes are* ***more than 6*** *then only record is moved to validKYCs*  *When upvotes are decreased to* ***less than 5*** *the record is removed from validKYCs*  ***Perform below command 6 times*** |

**Update KYC Votes**

truffle(development)> kyc.updateKYCVotes("sumeet",true,{from:accounts[1]})

**To See upvotes count**

truffle(development)> kyc.allCustomers(0)

**To See if record is moved to validKYCs**

truffle(development)> kyc.validKYCs(0)

|  |
| --- |
| ***Note:***  *Perform below command 3 times to make upvotes less than 5* |

**Updating KYC votes with false (down voting)**

truffle(development)> kyc.updateKYCVotes("sumeet",false,{from:accounts[1]})

**To See upvotes count**

truffle(development)> kyc.allCustomers(0)

**Other banks adding rating to bank**

truffle(development)> kyc.addRatingToBank(accounts[1])

**View Document hash of customer**

truffle(development)> kyc.viewCustomer("sumeet")

**Set Password to customer data**

truffle(development)> kyc.setCustomerPassword("sumeet","password",{from:accounts[1]})

**See the password set.**

truffle(development)> kyc.allCustomers(0)

**View Customer Rating provided by banks**

truffle(development)> await kyc.getCustomersRating("sumeet",{from:accounts[2]})

**View Bank Ratings**

truffle(development)> kyc.getBanksRating(accounts[1],{from:accounts[1]})

**Add Customer Ratings**

truffle(development)> kyc.addRatingToCustomer("sumeet",{from:accounts[1]})

**View Customer Ratings**

truffle(development)> kyc.getCustomersRating("sumeet",{from:accounts[1]})

**Remove KYC Request**

truffle(development)> kyc.removeRequest("sumeet",{from:accounts[1]})

**Remove Customer**

truffle(development)> kyc.removeCustomer("sumeet",{from:accounts[1]})

**Remove Bank by admin(contract owner)**

truffle(development)> kyc.removeBank(accounts[1])