**Build & Deploy smart contracts using Truffle**

**Setup Steps**

1. install node.js

 2.install truffle –

*$ npm install -g truffle*

  3. install testrpc

*$npm install -g ethereumjs-testrpc*

4. starts the private block chain

*$testrpc*

5. create a project directory

*$mkdir truffleproj*

*$cd truffleproj*

5. initialize your project

*$ truffle init*

This creates the required folder structure and some sample code

$ls -lrt

total 32

-rw-r--r-- 1 ninantharakan staff 154 Mar 1 20:19 truffle.js

drwxr-xr-x 3 ninantharakan staff 102 Mar 1 22:07 build

drwxr-xr-x 4 ninantharakan staff 136 Mar 2 14:41 test

drwxr-xr-x 4 ninantharakan staff 136 Mar 2 15:05 migrations

drwxr-xr-x 5 ninantharakan staff 170 Mar 2 22:24 contracts

contracts – store the contracts

build – store the compiled artifacts

test – store test scripts

migrations – store the deployment scripts

**Build & Deploy steps**

1. create your contracts in contracts directory

2. Compile all your contracts

*$ truffle compile*

This creates the compiled artifacts under build/contracts dir

3. Create the migration scripts in the migrations folder

4. Testing - test cases .sol scripts can be written in the test folder

5. Deploy all the contracts to testrpc blockchain . Use reset option if you want to deploy everything from beginning

$truffle migrate

Ninans-MacBook-Pro:truffleproj ninantharakan$ truffle migrate --reset

Using network 'development'.

Running migration: 1\_initial\_migration.js

  Replacing Migrations...

  Migrations: 0x2cbd79d124bab56199b91e0e7a22227d616ab66c

Saving successful migration to network...

Saving artifacts...

Running migration: 2\_deploy\_contracts.js

  Replacing ConvertLib...

  ConvertLib: 0xaf1a78e7f81ad93eb1417923f933dad166ab09ec

  Linking ConvertLib to MetaCoin

  Replacing MetaCoin...

  MetaCoin: 0x8b1b538fce94bf0931ebceb3def2427ba7071b6e

Saving successful migration to network...

Saving artifacts...

Ninans-MacBook-Pro:truffleproj ninantharakan$

**Interacting with contracts**

1. Open the truffle console

*$truffle console*

2. Get the info of contract MetaCoin

*truffle(development)> MetaCoin.deployed()*

3. Get a handle to the contract. Address of contract is available at the time of migration (Step 5 above)

*truffle(development)>*

*var meta=MetaCoin.at("0x8b1b538fce94bf0931ebceb3def2427ba7071b6e”);*

4. Invoke a contract function

*>meta.getBalance.call("0xd64b5af5a845fd5c49406cf906cc465beb77ba7d").then(console.log)*

output : { [String: '10000'] s: 1, e: 4, c: [ 10000 ] }

5. truffle(development)> meta.sendCoin("0x7af3c4cacdace66698b408a48a3dc80fb6a928a7",500);

Note: Call Vs Transaction

If the function makes any change to the blockchain you need to invoke the function without the call method

If you use call it’s a read only operation and doesn’t cost any gas

Web3.eth.contract() function

https://github.com/ethereum/wiki/wiki/JavaScript-API

Events :

<https://media.consensys.net/technical-introduction-to-events-and-logs-in-ethereum-a074d65dd61e#.iz4obodv3>

Get the list of accounts

truffle(development)> web3.eth.accounts

Help

truffle(development)>web3.eth

Get Ether balance for accounts

truffle(development)> web3.eth.getBalance(accounts[0]);

{ [String: '99927161900000000000'] s: 1, e: 19, c: [ 999271, 61900000000000 ] }

truffle(development)>

solidity tutorials

erisindustries.com

List of ethereum nodes

http://ethernodes.org/network/2

Exit truffle console

truffle(development)> .exit

send ether

<http://solidity.readthedocs.io/en/develop/types.html>

Sending Eth to a contract by calling the payable function

meta.sendEth({from:accounts[0],value:100});

UI Creation

1. npm install create-react-app
2. create a directory metacoinui and cd to it
3. create-react-app
4. npm start
5. update package.json – add web3 entry

MetaStock.sol

truffle(development)> meta.getStockBal.call(accounts[0]);

{ [String: '10000'] s: 1, e: 4, c: [ 10000 ] }

truffle(development)> meta.getStockBal.call(accounts[1]);

{ [String: '0'] s: 1, e: 0, c: [ 0 ] }

truffle(development)> meta.getEthBal.call(accounts[0]);

{ [String: '0'] s: 1, e: 0, c: [ 0 ] }

truffle(development)> meta.getEthBal.call(accounts[1]);

{ [String: '0'] s: 1, e: 0, c: [ 0 ] }

truffle(development)> web3.eth.getBalance(accounts[0]);

{ [String: '94831152599999999800'] s: 1, e: 19, c: [ 948311, 52599999999800 ] }

truffle(development)> web3.eth.getBalance(accounts[1]);

{ [String: '105000000000000000000'] s: 1, e: 20, c: [ 1050000 ] }

truffle(development)> web3.eth.getBalance(accounts[2]);

{ [String: '100000000000000000000'] s: 1, e: 20, c: [ 1000000 ] }

meta.buyMetaStock({from:accounts[1],value:100});

truffle(development)> meta.getEthBal.call(accounts[1]);

{ [String: '100'] s: 1, e: 2, c: [ 100 ] }

truffle(development)> meta.getEthBal.call(accounts[0]);

{ [String: '0'] s: 1, e: 0, c: [ 0 ] }

truffle(development)> meta.getStockBal.call(accounts[1]);

{ [String: '50'] s: 1, e: 1, c: [ 50 ] }

truffle(development)> meta.getStockBal.call(accounts[0]);

{ [String: '9950'] s: 1, e: 3, c: [ 9950 ] }

truffle(development)> web3.eth.getBalance(accounts[0]);

{ [String: '94831152599999999800'] s: 1, e: 19, c: [ 948311, 52599999999800 ] }

truffle(development)> web3.eth.getBalance(accounts[1]);

{ [String: '104993146299999999900'] s: 1, e: 20, c: [ 1049931, 46299999999900 ] }

truffle(development)> web3.eth.getBalance(meta.address);

{ [String: '100'] s: 1, e: 2, c: [ 100 ] }

meta.getEthBal.call(accounts[1]);

meta.getEthBal.call(accounts[0]);

meta.getStockBal.call(accounts[1]);

meta.getStockBal.call(accounts[0]);

web3.eth.getBalance(accounts[0]);

web3.eth.getBalance(accounts[1]);

web3.eth.getBalance(meta.address)

Overriding the default account in a function call.

function sellMetaStock(uint noofstocks) returns(bool)

meta.sellMetaStock(25, {from:accounts[1]}); /\* Default account is accounts[0]

**After selling 25 stocks account 1 have remaining 25 stocks & account 2 have 9975 stocks**

truffle(development)> meta.getStockBal.call({from: accounts[1]})

{ [String: '25'] s: 1, e: 1, c: [ 25 ] }

truffle(development)> meta.getStockBal.call()

{ [String: '9975'] s: 1, e: 3, c: [ 9975 ] }

**Checking the real ether balance of account and contract**

truffle(development)> web3.eth.getBalance(accounts[1]);

{ [String: '104486201899999999450'] s: 1, e: 20, c: [ 1044862, 1899999999450 ] }

truffle(development)> web3.eth.getBalance(meta.address);

{ [String: '150'] s: 1, e: 2, c: [ 150 ] }

**account 1 Buy some stocks**

meta.buyMetaStock({from:accounts[1],value:100});

**Checking account balance after the transaction**

truffle(development)> web3.eth.getBalance(accounts[1]);

{ [String: '104482348199999999350'] s: 1, e: 20, c: [ 1044823, 48199999999350 ] }

truffle(development)> web3.eth.getBalance(meta.address);

{ [String: '250'] s: 1, e: 2, c: [ 250 ] }

It means 100 ether / Wei got transferred to the contract.

**Check the stock balance**

truffle(development)> meta.getStockBal.call({from: accounts[1]});

{ [String: '125'] s: 1, e: 2, c: [ 125 ] }

**Sell some stocks**

truffle(development)> meta.sellMetaStock(25,{from:accounts[1]});

**Check the stock balance**

ruffle(development)> meta.getStockBal.call({from: accounts[1]});

{ [String: '100'] s: 1, e: 2, c: [ 100 ] }

**Checking account balance after the transaction**

truffle(development)> web3.eth.getBalance(accounts[1]);

{ [String: '104475748599999999400'] s: 1, e: 20, c: [ 1044757, 48599999999400 ] }

**Sell some stocks**

truffle(development)> meta.sellMetaStock(25,{from:accounts[1]});

gasUsed: 44371,

truffle(development)> web3.eth.getBalance(accounts[1]);

{ [String: '104471311499999999450'] s: 1, e: 20, c: [ 1044713, 11499999999450 ] }

<Since gas is reduced from account 1 we are not able to see the exact difference in ether bal>

But when we check the contract ether balance we can see that 100 wei is reduced ( we made the sell stock call 2 times with 25. i.e. 25\*2\*2 ( 1 stock is two eth)

truffle(development)> web3.eth.getBalance(meta.address);

{ [String: '150'] s: 1, e: 2, c: [ 150 ] }

**sell one stock**

truffle(development)> meta.sellMetaStock(1,{from:accounts[1]});

cumulativeGasUsed: 44371,

truffle(development)> web3.eth.getBalance(meta.address);

{ [String: '148'] s: 1, e: 2, c: [ 148 ] }

truffle(development)> web3.eth.getBalance(accounts[1]);

{ [String: '104466874399999999452'] s: 1, e: 20, c: [ 1044668, 74399999999452 ] }

**Change function to callable only**

function getStockBal() constant returns(uint) {

Links

<https://karl.tech/learning-solidity-part-1-deploy-a-contract/>

Online Remix IDE with Metamask

[*https://ethereum.github.io/browser-solidity*](https://ethereum.github.io/browser-solidity)