Ethereum

Part 3– Solidity Smart Contracts Tutorial

Prof. Marco Comuzzi
Department of Industrial Engineering
Ulsan National Institute of Science and Technology (UNIST)
mcomuzzi@unist.ac.kr

1. Ethereum Smart Contract Tutorial

Marco Comuzzi mcomuzzi@unistac.kr

Setup the development environment

Editor: remix.ethereum.org

Install Metamask plugin for your browser (Firefox, Chrome suggested)

Create accounts on Metamask, connect to Goerli network, get some GoeETH

Marco Comuzzi mcomuzzi@unistac.kr

Remix

File explorer: a simple editor (do frequent backups!)

Solidity Compiler

Deploy & Run Transactions

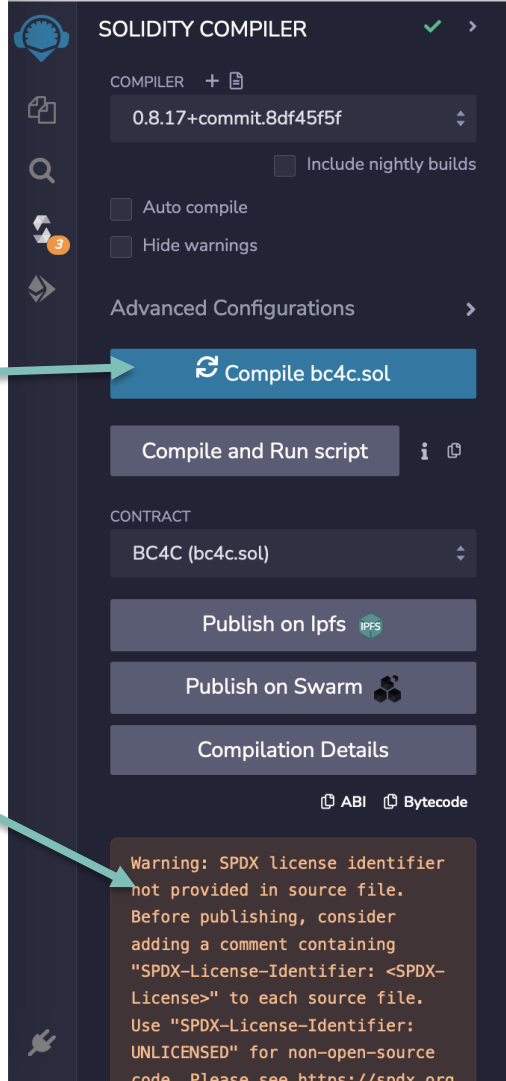
Marco Comuzzi mcomuzzi@unistac.kr

Solidity Compiler

Compile the contract currently opened in the file editor

Errors (red) and warnings (orange) shown below. You can ignore warnings.

Contract with errors cannot be deployed



Marco Comuzzi mcomuzzi@unistac.kr

Deploy & Run Transactions

Environment:

Remix VM (London): provides 10 "virtual" accounts with 100 ETH each. Transactions are immediately mined and included in blocks. There is no actual blockchain network

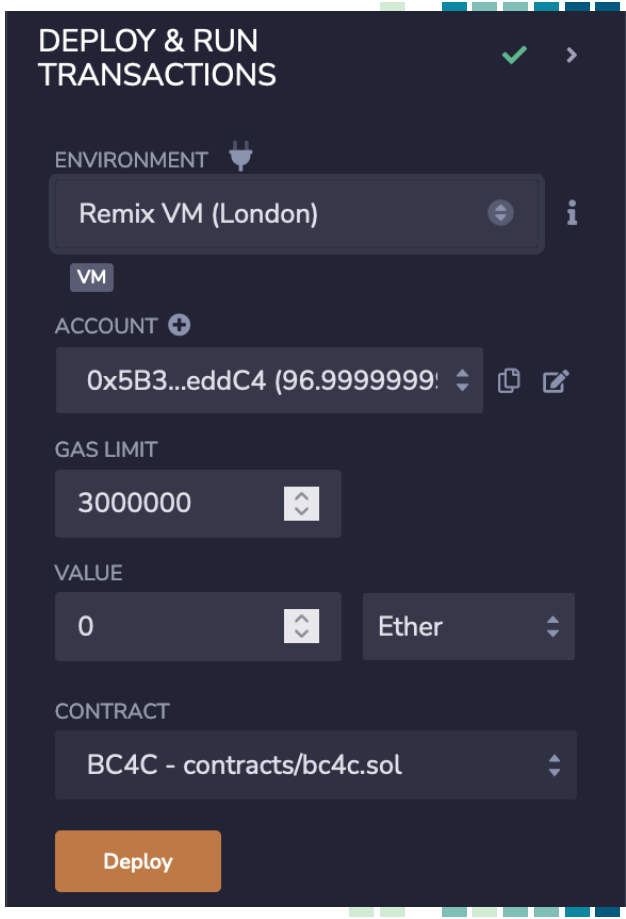
Injected Provider Metamask: if selected, the accounts are the ones that you have created in the Metamask wallet running in the same browser. Transactions submitted will be validated by nodes of the Goerli test network (and included forever in blocks!)

Gas limit (don't touch)

Value: the value transferred by the next transaction issued

Contract: the contract under consideration

Deploy: click here to deploy
If Metamask environment selected, you will have to confirm every transaction in Metamask



Marco Comuzzi mcomuzzi@unistac.kr

Deploy & Run Transactions

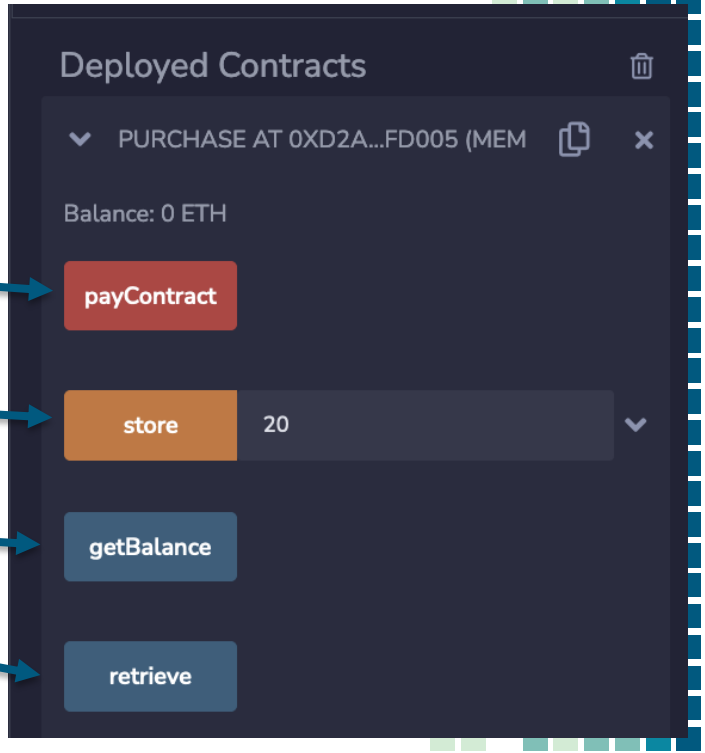
For each deployed contract the functions are listed:

Payable

Other public (not payable, but that modify the state)

View

Can be invoked by specifying the parameters (click on "transact")



Marco Comuzzi mcomuzzi@unistac.kr

```
    // data of the contract (state variable)
    uint number;

    // called once when the contract is deployed
    constructor() public {
        number = 0;
    }

    // change the value of the stored number
    function store(uint _number) public {
        number = _number;
    }

    // return the value of the number cuurently stored (
    // Note: "view" because it does not modify the state
    function retrieve() public view returns (uint){
        return number;
    }

    // The ETH paid are the ones in the "value" field of the invoking transaction
    function payContract() public payable{
```

contract Purchase

0 listen on all transactions Search with transaction hash or address

```
call to Purchase.getBalance

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Purchase.getBalance() data: 0x120...65fe0
transact to Purchase.payContract pending ...

[vm] from: 0x5B3...eddC4 to: Purchase.payContract() 0xDA0...42B53 value: 3000000000000000000 wei data: 0xecb...0b862 logs: 0
hash: 0xe3a...9ed8e
call to Purchase.getBalance

CALL [call] from: 0x5B38Da6a701c568545dCfcB
creation of Purchase pending...

[vm] from: 0x5B3...eddC4 to: Purchase.(constructor) value: 0 wei data: 0x608...10033 logs: 0 hash: 0xa09...6c8e7
```

Debug Debug Debug Debug

Effects of transactions seen here, click on debug to see more details. All transactions and blocks can also be seen on Etherscan (if using Metamask connected to Goerli Test Network)

demo.sol
(simple contract with no particular meaning)

Called only once when the contract is deployed

Public function that changes the value of the number stored (changes the state)

Retrieve the value stored (”view” >> does not change the state)

An empty payable function: the ETH in the “value” field of the tx invoking this function will be transferred to the contract

Returns the balance of the caller (note the use of the special variable to ref. the balance of the originator of the tx invoking this function

```
contract Purchase {

    // data of the contract (state variable)
    uint number;

    // called once when the contract is deployed
    constructor() public {
        number = 0;
    }

    // change the value of the stored number
    function store(uint _number) public {
        number = _number;
    }

    // return the value of the number cuurently stored (
    // Note: "view" because it does not modify the state
    function retrieve() public view returns (uint){
        return number;
    }

    // The ETH paid are the ones in the "value" field of
    //the invoking transaction
    function payContract() public payable{

    }

    // Returns the ETH balance of this contract
    function getBalance() public view returns (uint256){
        return address(this).balance;
    }
```

purchase.sol

A smart contract to purchase an object using cryptocurrency

Marco Comuzzi mcomuzzi@unistac.kr

bc4c.sol

An implementation of the BC4C-CGM blockchain

Instead of "CGM tokens", Ether (ETH) is used to pay the deposits

Marco Comuzzi mcomuzzi@unistac.kr

ubstv2.sol

An ERC-20 token smart contract

Only one user (Bob) can initialize it, creating the initial supply

Marco Comuzzi mcomuzzi@unistac.kr

ubstvopen.sol

An ERC-20 token smart contract

Any node can increase (mint new tokens) or decrease (burn existing tokens) the token supply

Marco Comuzzi mcomuzzi@unistac.kr

Nft.sol

A basic example of an ERC-721 Non-functional token

Marco Comuzzi mcomuzzi@unistac.kr

THANKS!

https://sites.google.com/site/marcocomuzziphd

http://iel.unist.ac.kr/

You can find me at:

@dr_bsad

mcomuzzi@unist.ac.kr