BUILDING DEFI PROTOCOL ERC20 Token Smart Contract for Ethereum Blockchain

Since the inception of Blockchain technology; Bitcoin, Ethereum, or crypto-currencies are hot topics and buzzing around the world many startups based on Blockchain technologies are using cryptocurrencies, in other words, crypto tokens for the utilization of their products. These crypto tokens can be deployed on many Blockchain like Ethereum, Cardano, Binance, Polkadot, etc. It's another topic of discussion, on which blockchain these crypto tokens need to be implemented but as Ethereum being the first market mover, this blog post explains, how you can create such a token on the Ethereum blockchain.

Before creating an Ethereum based token (ERC20 token), understand first the basics of Smart-contract and their native programming language Solidity.

Smart Contract

A smart contract is simply a set of rules that contains the business logic or a protocol according to which all the transactions on a Blockchain should happen. The general purpose of a Smart contract is to satisfy common contractual conditions like creating its token, perform arbitrary competitions, function to send and receive tokens, and store states of transactions.

Solidity

Solidity is an object-oriented and high-level smart-contract programming language, which is developed on top of Ethereum Virtual Machine (EVM). Solidity compiler converts smart-contract code into EVM bytecode which is sent to the Ethereum network as a deployment transaction. It would be best to have a good understanding of Solidity programming language to efficiently write an Ethereum Smart Contract and build an application on smart-contract.
Coding example of smart-contract

This section contains the example of a smart-contract code written using the Solidity programming language.

Prerequisite

Integrated development environment (IDE)

Remix as the IDE. It is a web-based IDE with built-in static analysis and a testnet EVM. Remix provides the possibility to compile and deploy it to Ethereum testnet with Metamask. Here is a good blog post for it.

There is also another web-based IDE available like EthFiddle. For more information related to IDE please visit here.

Programming Language

Solidity

ERC20 Token Info

- Symbol – **N47**
- Name – **N47Token**
- Decimals – **0**
- Total Supply – **1000000**

Smart-contract Code

```
// SPDX-License-Identifier: unlicensed
pragma solidity 0.8.4;
//----------------------------------------------------------------------
// Safe maths
//----------------------------------------------------------------------
contract SafeMath {
    function safeAdd(uint a, uint b) public pure returns (uint c) {
        c = a + b;
        require(c >= a);
    }
    function safeSub(uint a, uint b) public pure returns (uint c) {
        require(b <= a);
        c = a - b;
    }
}
//----------------------------------------------------------------------
// ERC Token Standard #20 Interface
```

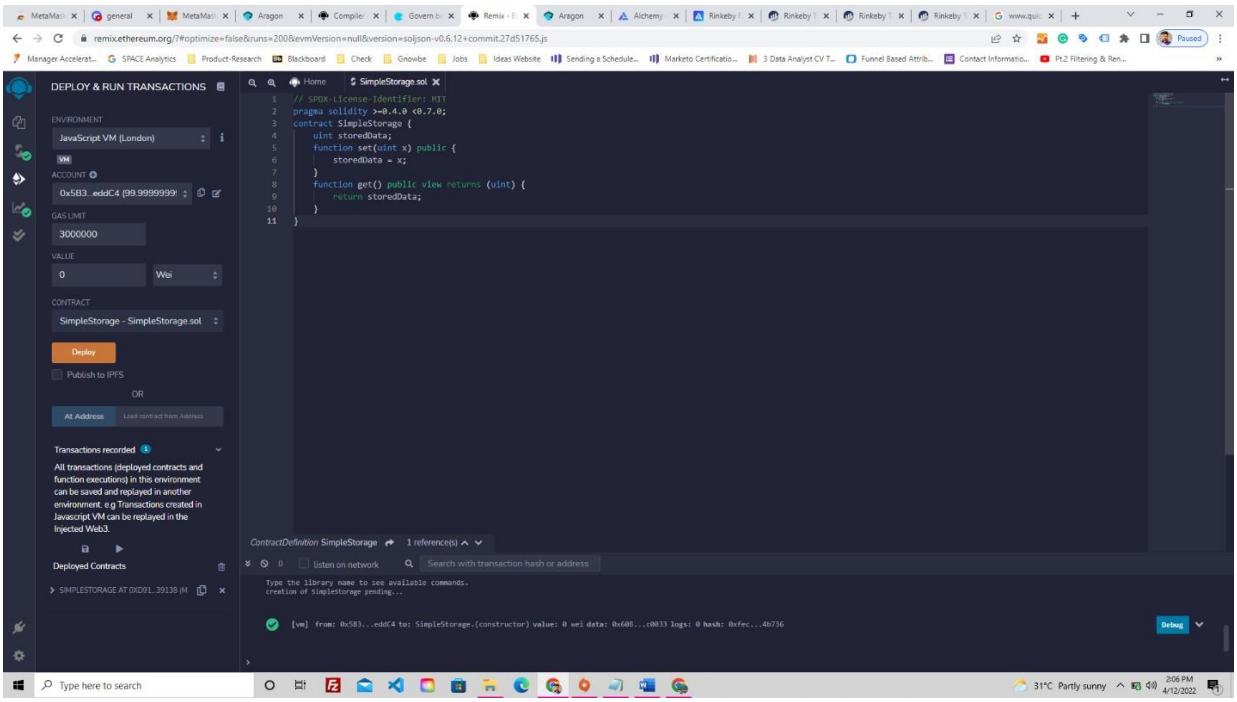
```
18// https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
19//----------------------------------------------------------------------
20abstract contract ERC20Interface {
21    function totalSupply() virtual public view returns (uint);
22    function balanceOf(address tokenOwner) virtual public view returns (uint balance);
23    function allowance(address tokenOwner, address spender) virtual public view returns (uint remaining);
24    function transfer(address to, uint tokens) virtual public returns (bool success);
25    function approve(address spender, uint tokens) virtual public returns (bool success);
26    function transferFrom(address from, address to, uint tokens) virtual public returns (bool success);
27    event Transfer(address indexed from, address indexed to, uint tokens);
28    event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
29}
30//----------------------------------------------------------------------
31// ERC20 Token, with the addition of symbol, name and decimals
32// assisted token transfers
33//----------------------------------------------------------------------
34contract N47Token is ERC20Interface, SafeMath {
35    string public symbol;
36    string public  name;
37    uint8 public decimals;
38    uint public _totalSupply;
39    mapping(address => uint) balances;
40    mapping(address => mapping(address => uint)) allowed;
41    // ------------------------------------------------------------------------
42    // Constructor
43    // ------------------------------------------------------------------------
44    constructor() {
45        symbol = "N47";
```

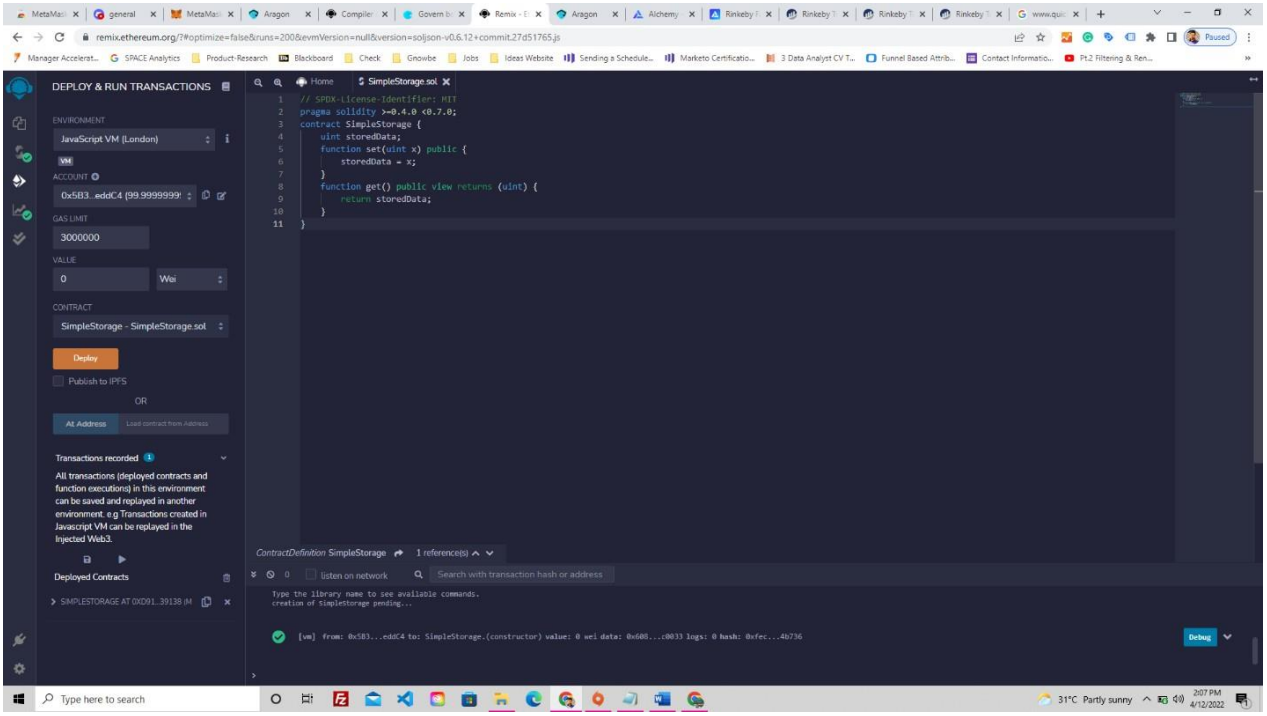
```
46      name = "N47Token";
47      decimals = 0;
48      _totalSupply = 1000000;
49      balances[msg.sender] = _totalSupply;
50      emit Transfer(address(0), msg.sender, _totalSupply);
51  }
52  // ------------------------------------------------------------------------
53  // Total supply
54  // ------------------------------------------------------------------------
55  function totalSupply() public override view returns (uint) {
56  return _totalSupply - balances[address(0)];57
    }
58  // ------------------------------------------------------------------------
59  // Get the token balance for account tokenOwner
60  // ------------------------------------------------------------------------
61  function balanceOf(address tokenOwner) public override view returns (uint balance) {
62  return balances[tokenOwner];63
    }
64  // ------------------------------------------------------------------------
65  // Transfer the balance from token owner's account to receiver account
66  // - Owner's account must have sufficient balance to transfer
67  // - 0 value transfers are allowed
68  // ------------------------------------------------------------------------
69  function transfer(address receiver, uint tokens) public override returns (bool success) {
70      balances[msg.sender] = safeSub(balances[msg.sender], tokens);
71      balances[receiver] = safeAdd(balances[receiver], tokens);
72      emit Transfer(msg.sender, receiver, tokens);
73      return true;
```

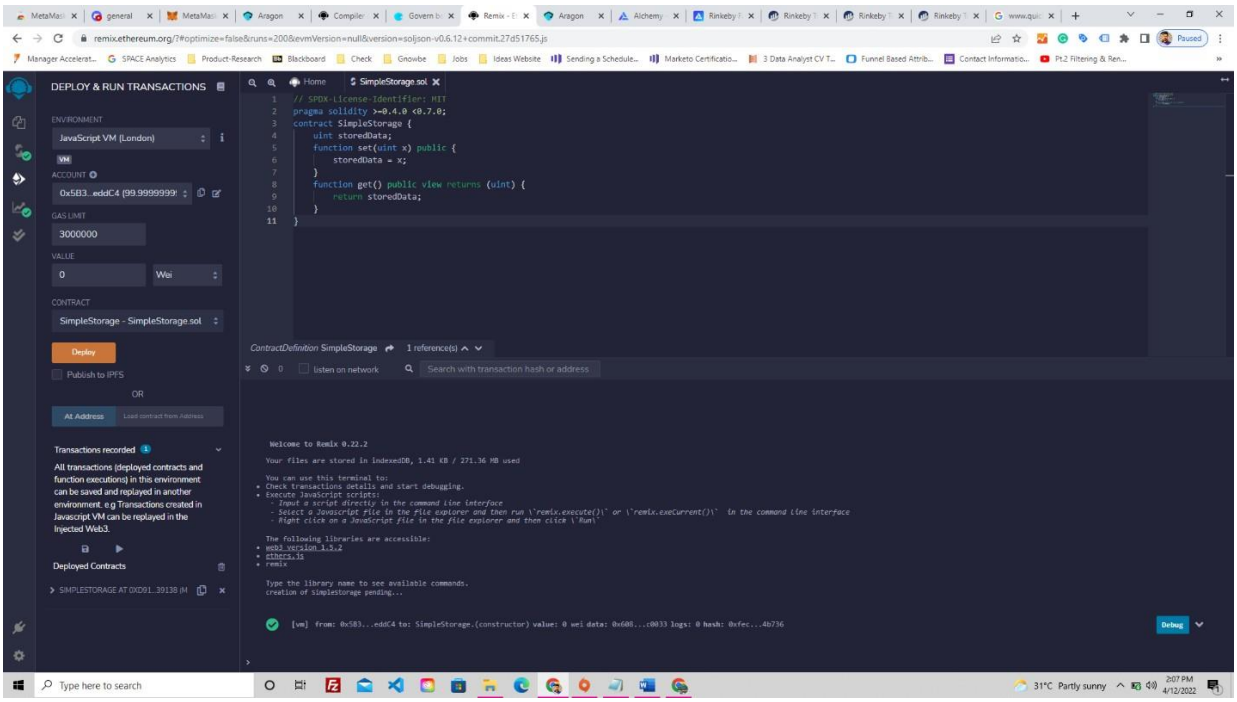
```
74  }
75  // ----------------------------------------------------------------------
76  // Token owner can approve for spender to transferFrom(...) tokens
77  // from the token owner's account
78  //
79  // https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
80  // recommends that there are no checks for the approval double-spend attack
81  // as this should be implemented in user interfaces
82  // ----------------------------------------------------------------------
83  function approve(address spender, uint tokens) public override returns (bool success) {
84      allowed[msg.sender][spender] = tokens;
85      emit Approval(msg.sender, spender, tokens);
86  return true;87
    }
88  // ----------------------------------------------------------------------
89  // Transfer tokens from sender account to receiver account
90  //
91  // The calling account must already have sufficient tokens approve(...)-d
92  // for spending from sender account and
93  // - From account must have sufficient balance to transfer
94  // - Spender must have sufficient allowance to transfer
95  // - 0 value transfers are allowed
96  // ----------------------------------------------------------------------
97  function transferFrom(address sender, address receiver, uint tokens) public override returns (bool success
98      balances[sender] = safeSub(balances[sender], tokens);
99      allowed[sender][msg.sender] = safeSub(allowed[sender][msg.sender], tokens);
100     balances[receiver] = safeAdd(balances[receiver], tokens);
101     emit Transfer(sender, receiver, tokens);
```

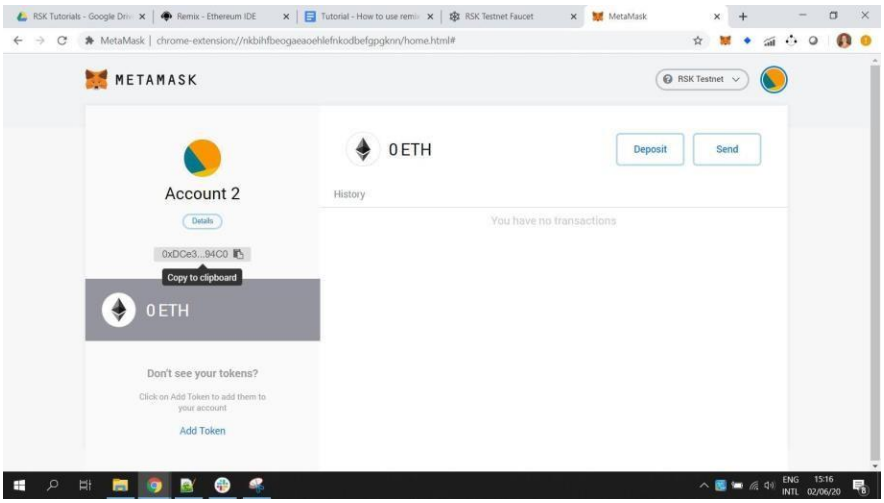
```
102         return true;
103     }
104     // ----------
105     // Returns the amount of tokens approved by the owner that can be
106     // transferred to the spender's account
107     // ----------
108     function allowance(address tokenOwner, address spender) public override view returns (uint remaining) {
109         return allowed[tokenOwner][spender];
110     }
111 }
```

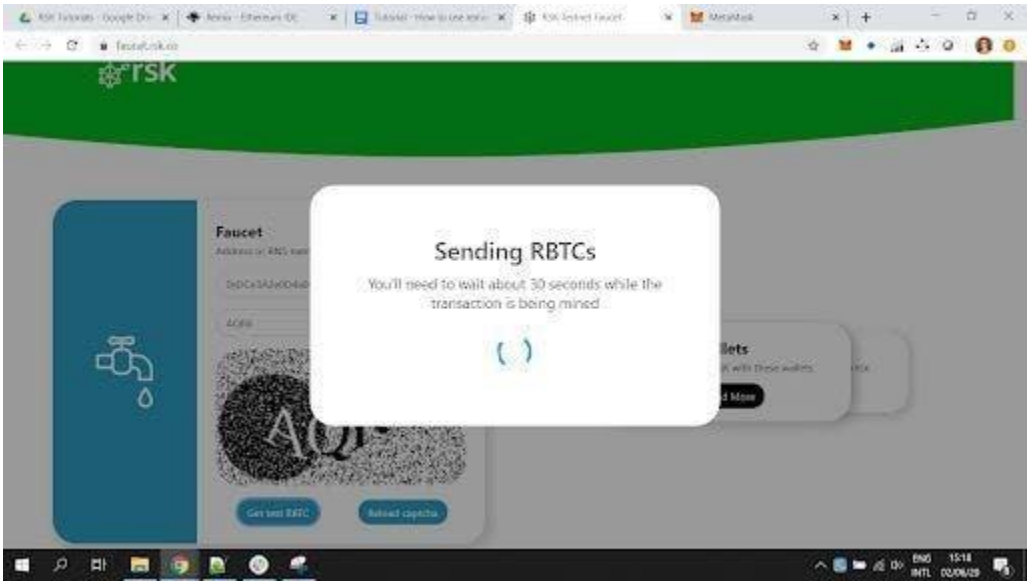
Using the above code, smart-contract can be deployed on Ethereum Mainnet or Testnet. Deploying a smart contract is technically a transaction, that needs to pay Gas (fees) in terms of ETH (Native token for Ethereum network), in the same way, that needs to pay gas for a simple ETH transfer. However, Gas costs for contract deployment are far higher.



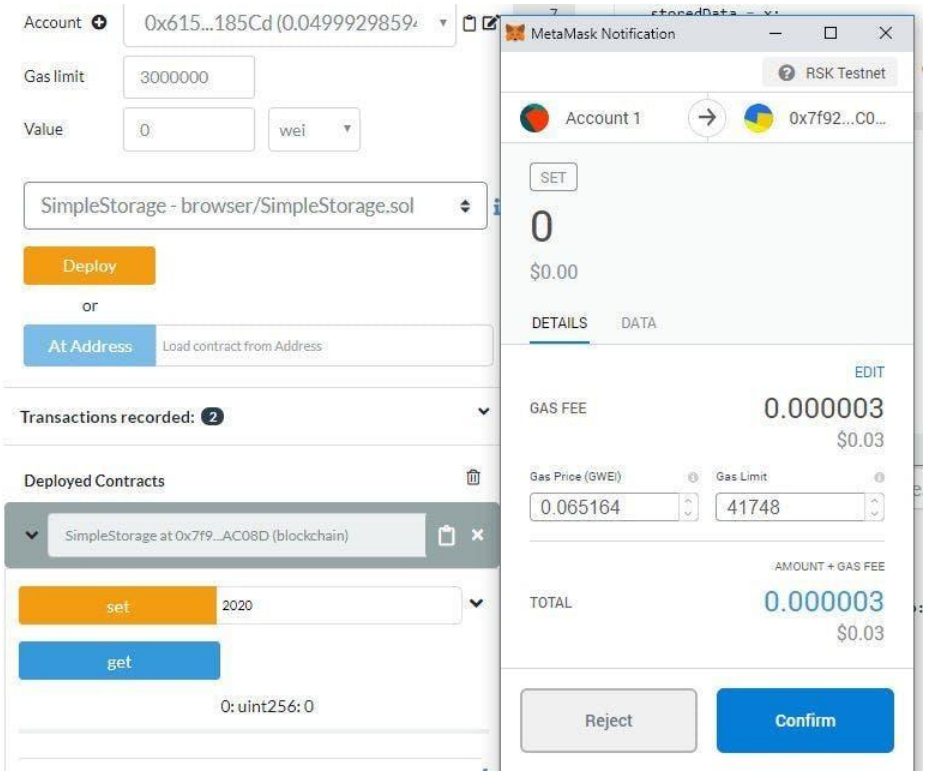

DEPLOY & RUN TRANSACTIONS
ENVIRONMENT
JavaScript VM (London)
ACCOUNT
0x5B3...eddC4 (99.99999999
GAS LIMIT
3000000
VALUE
0
Wei
CONTRACT
SimpleStorage - SimpleStorage.sol
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded
All transactions (deployed contracts and function executions) in this environment can be saved and replayed in another environment. e.g Transactions created in Javascript VM can be replayed in the Injected Web3.
Deployed Contracts
SIMPLESTORAGE AT 0XD91...39138 (M
Home
SimpleStorage.sol
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.0 <0.7.0;
contract SimpleStorage {
uint storedData;
function set(uint x) public {
storedData = x;
}
function get() public view returns (uint) {
return storedData;
}
}
ContractDefinition SimpleStorage
1 reference(s)
listen on network
Search with transaction hash or address
Type the library name to see available commands.
creation of simplestorage pending...
[vm] from: 0x5B3...eddC4 to: SimpleStorage.(constructor) value: 0 wei data: 0x608...c0033 logs: 0 hash: 0xfec...4b736
Debug
Type here to search
31°C Partly sunny
2:07 PM
4/12/2022

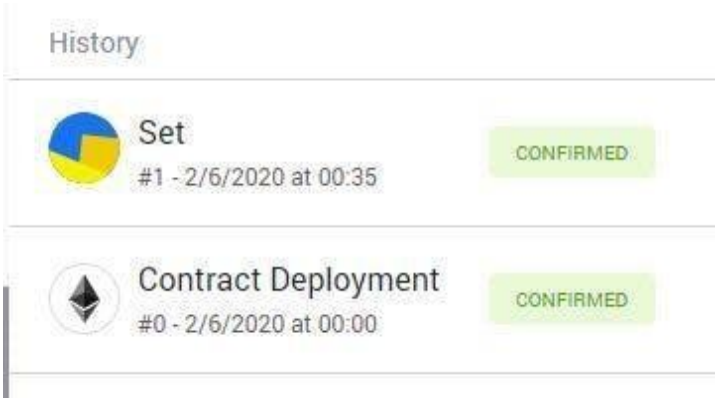

DEPLOY & RUN TRANSACTIONS
ENVIRONMENT
JavaScript VM (London)
ACCOUNT
0x5B3...eddC4 (99.99999999
GAS LIMIT
3000000
VALUE
0
Wei
CONTRACT
SimpleStorage - SimpleStorage.sol
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded
All transactions (deployed contracts and function executions) in this environment can be saved and replayed in another environment. e.g Transactions created in Javascript VM can be replayed in the Injected Web3.
Deployed Contracts
SIMPLESTORAGE AT 0XD91...39138 (M
Home
SimpleStorage.sol
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.0 <0.7.0;
contract SimpleStorage {
uint storedData;
function set(uint x) public {
storedData = x;
}
function get() public view returns (uint) {
return storedData;
}
}
ContractDefinition SimpleStorage
1 reference(s)
listen on network
Search with transaction hash or address
Welcome to Remix 0.22.2
Your files are stored in indexedDB, 1.41 KB / 271.36 MB used
You can use this terminal to:
Check transactions details and start debugging.
Execute JavaScript scripts:
Input a script directly in the command line interface
Select a Javascript file in the file explorer and then run \"remix.execute()\" or \"remix.exeCurrent()\" in the command line interface
Right click on a JavaScript file in the file explorer and then click \"Run\"
The following libraries are accessible:
web3 version 1.5.2
ethers.js
remix
Type the library name to see available commands.
creation of simplestorage pending...
[vm] from: 0x5B3...eddC4 to: SimpleStorage.(constructor) value: 0 wei data: 0x608...c0033 logs: 0 hash: 0xfec...4b736
Debug
Type here to search
31°C Partly sunny
2:07 PM
4/12/2022


METAMASK
RSK Testnet
Account 2
Details
0xDCe3...94C0
Copy to clipboard
0 ETH
0 ETH
Deposit
Send
History
You have no transactions
Don't see your tokens?
Click on Add Token to add them to your account
Add Token


rsk
Faucet
Sending RBTCs
You'll need to wait about 30 seconds while the transaction is being mined


Sent
Successfully sent some RBTCs to
0x61545F123fa9d73f6eBCB940975EcdcF81d185Cd
Transaction hash
OK


Account
0x615...185Cd (0.0499929859
Gas limit
3000000
Value
0
wei
SimpleStorage - browser/SimpleStorage.sol
Deploy
or
At Address
Load contract from Address
Transactions recorded: 2
Deployed Contracts
SimpleStorage at 0x7f9...AC08D (blockchain)
set
2020
get
0: uint256: 0
MetaMask Notification
RSK Testnet
Account 1
0x7f92...C0...
SET
0
$0.00
DETAILS
DATA
EDIT
GAS FEE
0.000003
$0.03
Gas Price (GWEI)
0.065164
Gas Limit
41748
AMOUNT + GAS FEE
TOTAL
0.000003
$0.03
Reject
Confirm


History
Set
#1 - 2/6/2020 at 00:35
CONFIRMED
Contract Deployment
#0 - 2/6/2020 at 00:00
CONFIRMED