CREATE A CUSTOM TOKEN AND DEPLOY IT ON ROPSTEN NETWORK

Abstract

This book chapter serves as a practical guide for individuals keen on understanding the process of creating a custom ERC-20 token and deploying it on the Ropsten network using MetaMask. The chapter delves into the foundational knowledge required to comprehend blockchain technology, Ethereum, smart contracts, and the significance of custom tokens in the decentralized ecosystem.

Subsequently, the chapter provides a step-by-step tutorial on creating a custom token using Solidity, a programming language for writing smart contracts on the Ethereum blockchain.

The practical guide transitions into elucidating the process of deploying the custom token on the Ropsten test network using MetaMask, a popular Ethereum wallet and gateway to blockchain applications. It covers setting up MetaMask, obtaining test ether, compiling and deploying the smart contract, and verifying the deployment on the Ropsten network through Etherscan.

Tools Required: Ubuntu, web browser, Remix IDE

Prerequisites: Metamask, Test Faucet, Etherscan

Keywords: Custom, Token, Deploy, Ropsten.

Authors

Pathan Mahamood Khan
Assistant Professor
Department of Electrical and Electronics Engineering
Vasireddy Venkatadri Institute of Technology, Nambur, Andhra Pradesh.
pathanmehemudkhan@gmail.com

Shaik Mulla Almas
Assistant Professor
Department of Information Technology
Vasireddy Venkatadri Institute of Technology, Nambur, Andhra Pradesh.
mullaalmas27@gmail.com

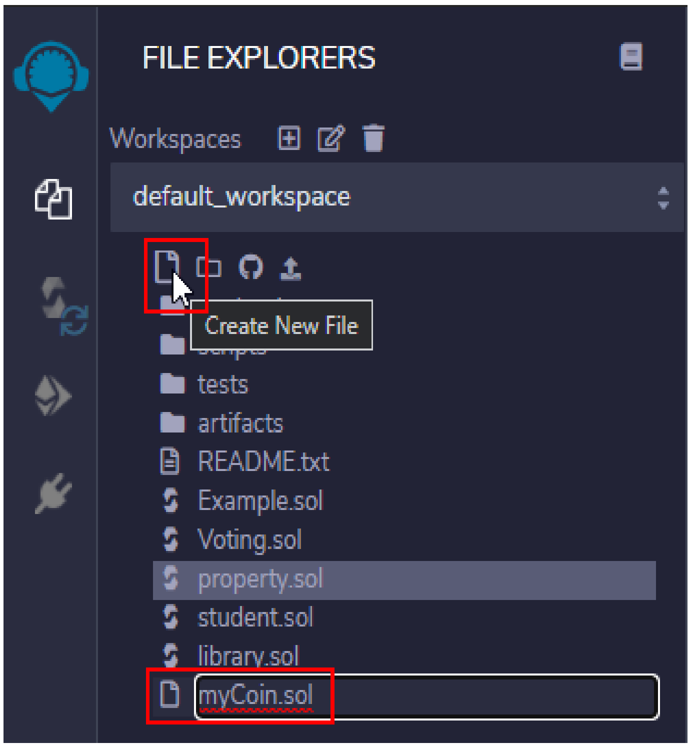
Rameshkumar Jayaraman
Assistant Professor
Department of Electrical Engineering
Annamalai University
Chidambaram, Tamil Nadu.
rameshwin75@gmail.com

I. STEPS TO BE FOLLOWED

1. Creating the custom token smart contract using the text editor
2. Compiling the contract using Injected Web3 and Metamask
3. Deploying the contract on the Ropsten network using Metamask

Step 1: Creating the Custom Token Smart Contract using the Text Editor

- Open Remix IDE and click the **Create new file** button to create a **myCoin.sol** file



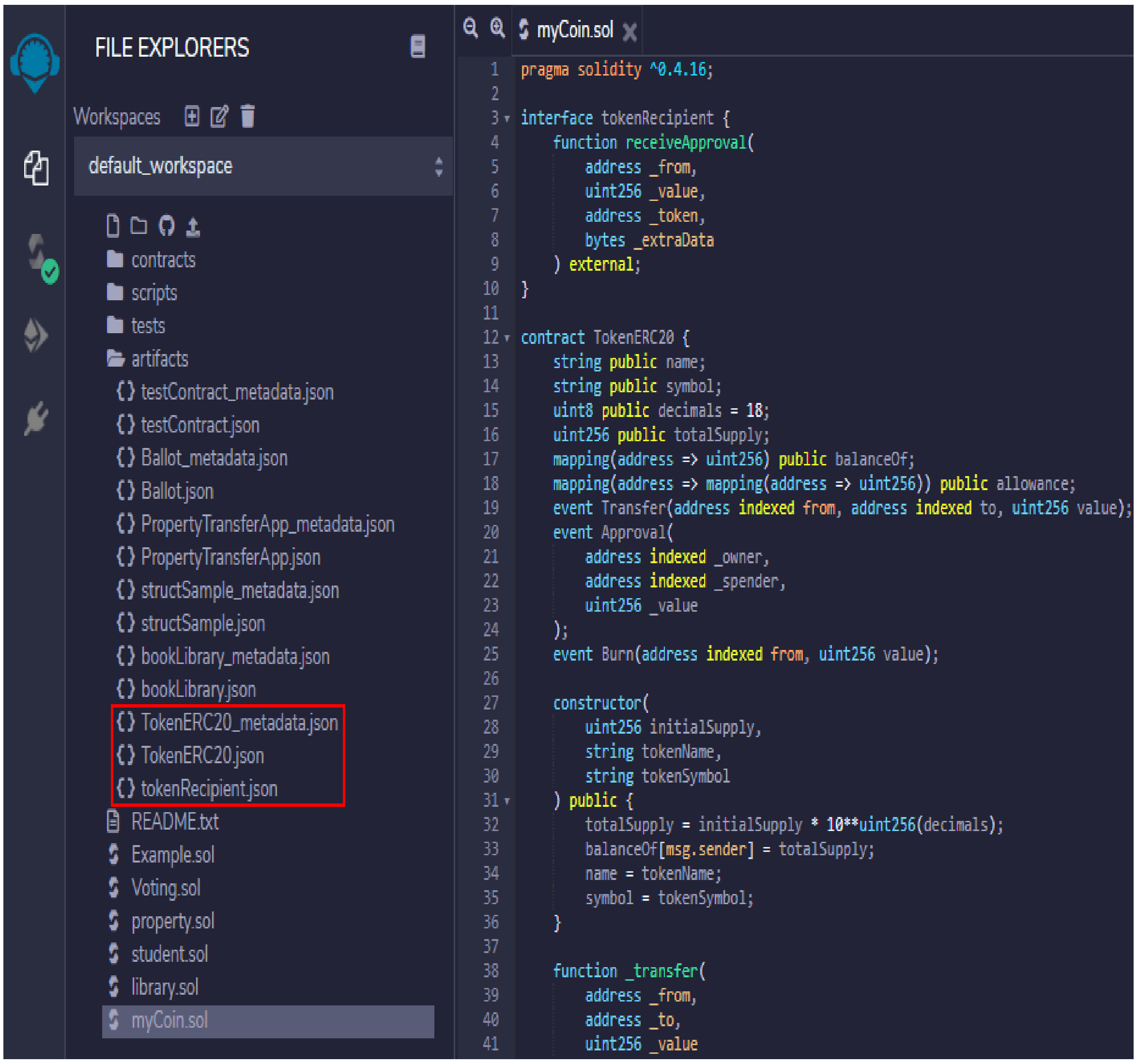
- Once the file is created, type the following code in the **myCoin.sol**file

```
pragma solidity ^0.4.16;

interface tokenRecipient {
   function receiveApproval(
      address  _from,
      uint256
      _value,
      address
      _token,     bytes
      _extraData
   ) external;
}

contract TokenERC20
   { string public
```

```
    name; string public
    symbol;
    uint8 public decimals = 18;
    uint256 public totalSupply;
mapping(address => uint256) public balanceOf;
mapping(address => mapping(address => uint256)) public allowance;
    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(
        address indexed _owner,
        address indexed
        _spender, uint256
        _value
    );
    event Burn(address indexed from, uint256 value);

constructor(
        uint256
        initialSupply, string
        tokenName, string
        tokenSymbol
    ) public {
totalSupply = initialSupply * 10**uint256(decimals);
balanceOf[msg.sender] = totalSupply;
        name = tokenName;
        symbol = tokenSymbol;
    }

    function
        _transfer(
        address _from,
        address _to,
        uint256 _value
    ) internal {
require(_to != 0x0); require(balanceOf[_from] >=
        _value); require(balanceOf[_to] + _value >=
        balanceOf[_to]);
        uint256 previousBalances = balanceOf[_from] + balanceOf[_to];
balanceOf[_from] -= _value;
balanceOf[_to] += _value;
        emit Transfer(_from, _to, _value);
        assert(balanceOf[_from] + balanceOf[_to] == previousBalances);
    }

    function transfer(address _to, uint256 _value)
        public returns (bool success)
    {
        _transfer(msg.sender, _to, _value);
        return true;
    }
```

```
  function
     transferFrom(
     address _from,
     address _to,
     uint256 _value
  ) public returns (bool success) {
require(_value <= allowance[_from][msg.sender]); // Check allowance
     allowance[_from][msg.sender] -= _value;
     _transfer(_from, _to, _value);
     return true;
  }

  function approve(address _spender, uint256 _value)
     public returns (bool success)
  {
     allowance[msg.sender][_spender] = _value;
     emit Approval(msg.sender, _spender,
     _value); return true;
  }

  function
     approveAndCall(
     address _spender,
     uint256 _value,
     bytes _extraData
  ) public returns (bool success) {
tokenRecipient spender = tokenRecipient(_spender);
     if (approve(_spender, _value)) {
spender.receiveApproval(msg.sender, _value, this, _extraData);
          return
           true;
     }
  }

  function burn(uint256 _value) public returns (bool success) {
     require(balanceOf[msg.sender] >= _value);
balanceOf[msg.sender] -= _value;
totalSupply -= _value;
     emit Burn(msg.sender, _value);
     return true;
  }

  function burnFrom(address _from, uint256 _value)
     public returns (bool success)
  {
     require(balanceOf[_from] >= _value);
require(_value <= allowance[_from][msg.sender]);
balanceOf[_from] -= _value;
     allowance[_from][msg.sender] -= _value;
totalSupply -= _value;
        emit Burn(_from, _value);
        return true;
    }
}
```

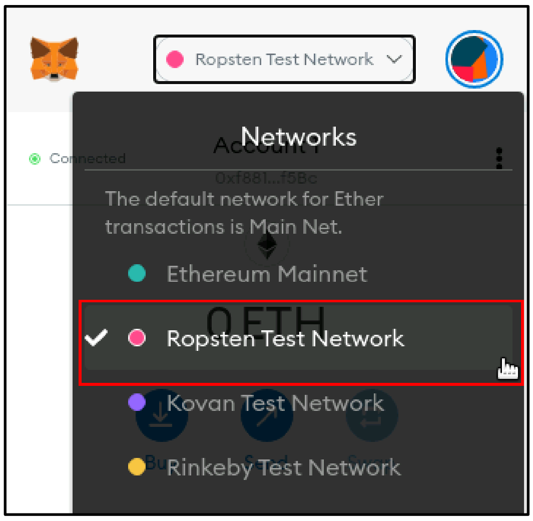
- Once the smart contract code is typed, save the file. Once the **myCoin.sol** file is saved, observe that JSON metadata will be created for the file



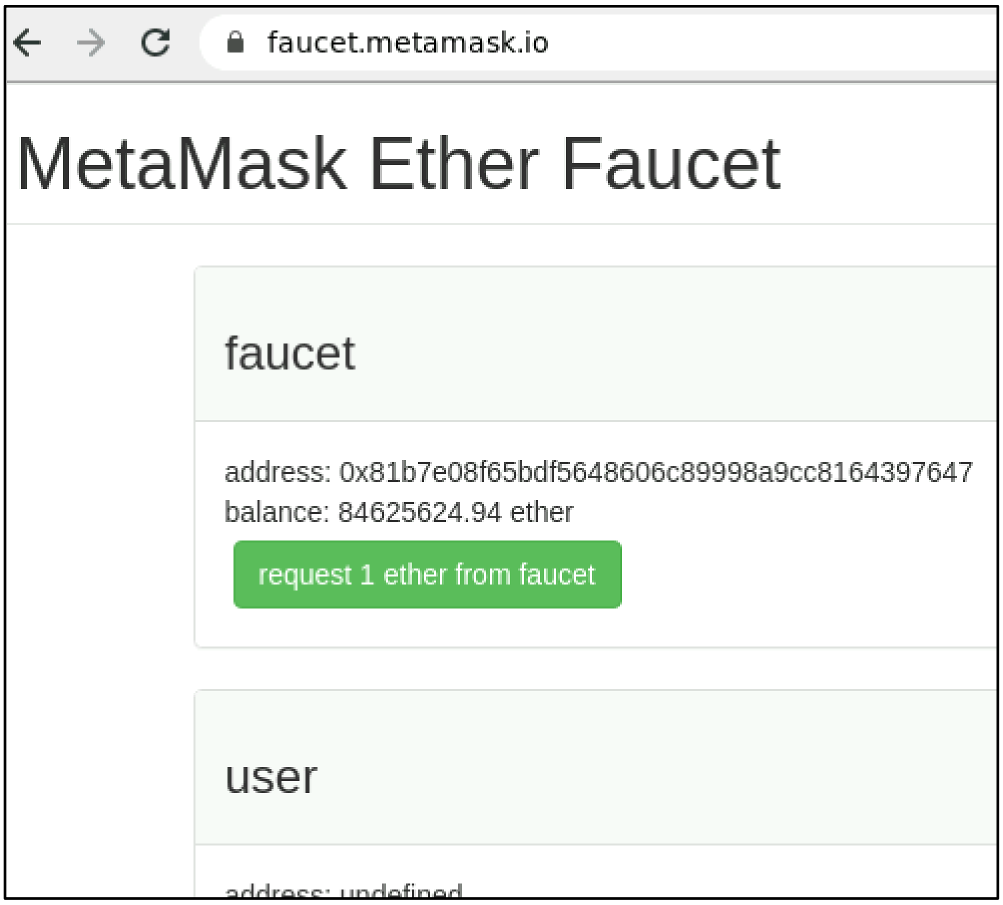
```
pragma solidity ^0.4.16;

interface tokenRecipient {
    function receiveApproval(
        address _from,
        uint256 _value,
        address _token,
        bytes _extraData
    ) external;
}

contract TokenERC20 {
    string public name;
    string public symbol;
    uint8 public decimals = 18;
    uint256 public totalSupply;
    mapping(address => uint256) public balanceOf;
    mapping(address => mapping(address => uint256)) public allowance;
    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(
        address indexed _owner,
        address indexed _spender,
        uint256 _value
    );
    event Burn(address indexed from, uint256 value);

    constructor(
        uint256 initialSupply,
        string tokenName,
        string tokenSymbol
    ) public {
        totalSupply = initialSupply * 10**uint256(decimals);
        balanceOf[msg.sender] = totalSupply;
        name = tokenName;
        symbol = tokenSymbol;
    }

    function _transfer(
        address _from,
        address _to,
        uint256 _value
```

Step 2: Compiling the Contract using Injected Web3 and Metamask

- Connect to the **Ropsten Test Network** by clicking on the button in the **Networks** Menu



- Open the Metamask Ether Faucet by navigating to the following website: https://faucet.metamask.io


faucet.metamask.io

MetaMask Ether Faucet

faucet

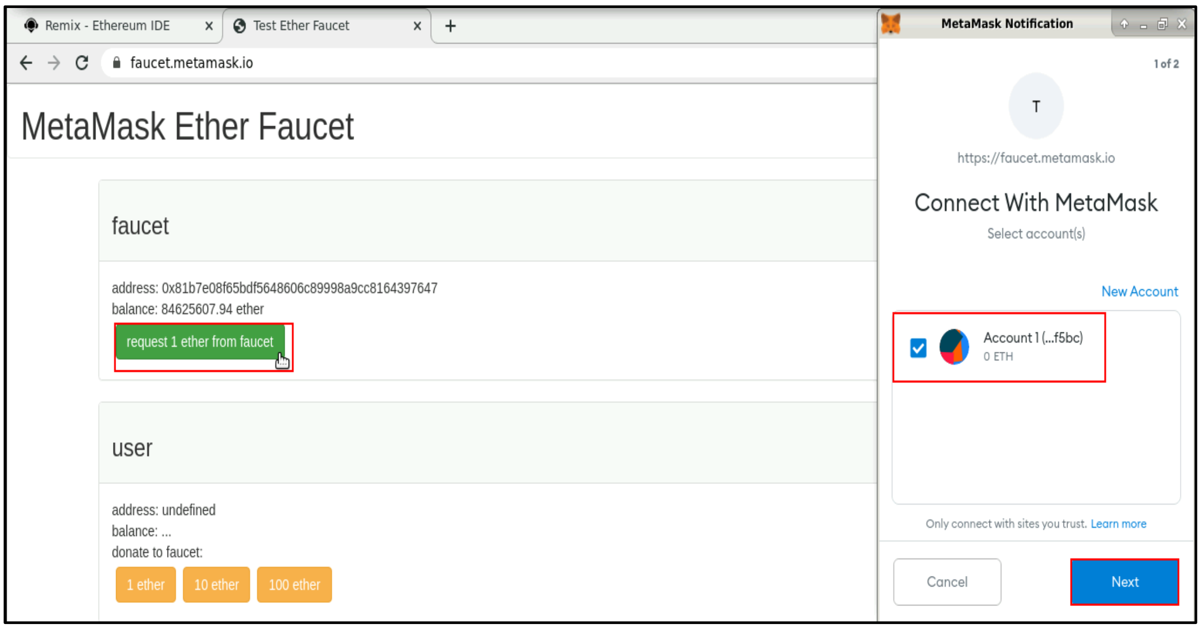
address: 0x81b7e08f65bdf5648606c89998a9cc8164397647
balance: 84625624.94 ether

request 1 ether from faucet

user

address: undefined

- Click on the **Request 1 Ether from faucet** button which prompts Metamask to select an account to receive Ether. Select the account of your choice and click **Next**



MetaMask Ether Faucet

faucet

address: 0x81b7e08f65bdf5648606c89998a9cc8164397647
balance: 84625607.94 ether

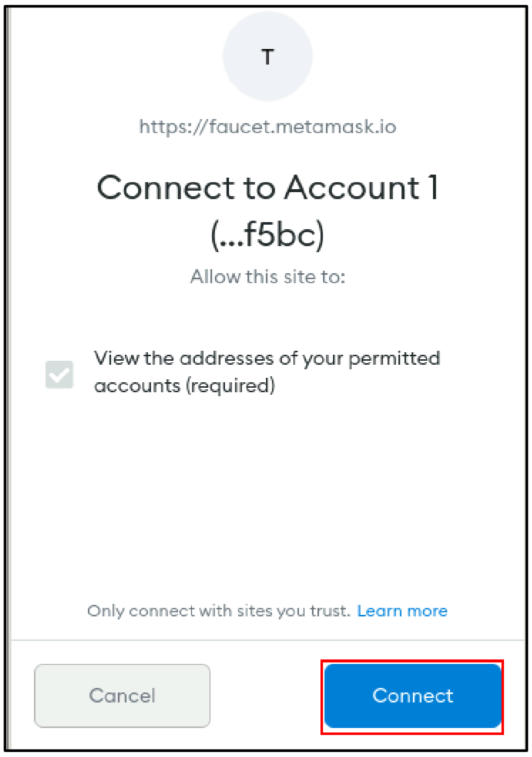
request 1 ether from faucet

user

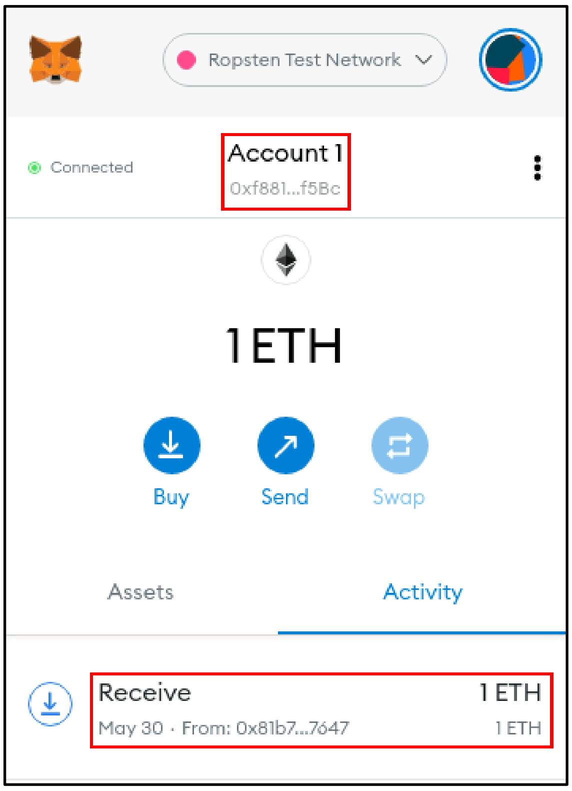
address: undefined
balance: ...
donate to faucet:

1 ether 10 ether 100 ether

- Click the **Connect** button to connect to the faucet and receive Ether

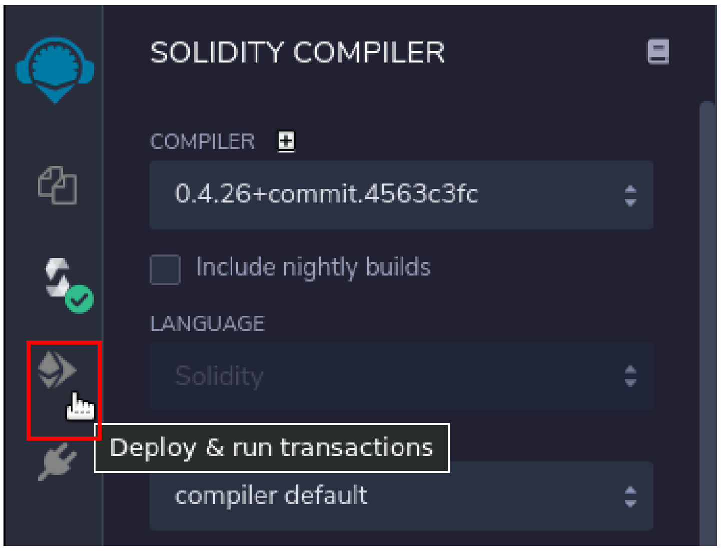

T
https://faucet.metamask.io
Connect to Account 1 (...f5bc)
Allow this site to:
View the addresses of your permitted accounts (required)
Only connect with sites you trust. Learn more
Cancel
Connect

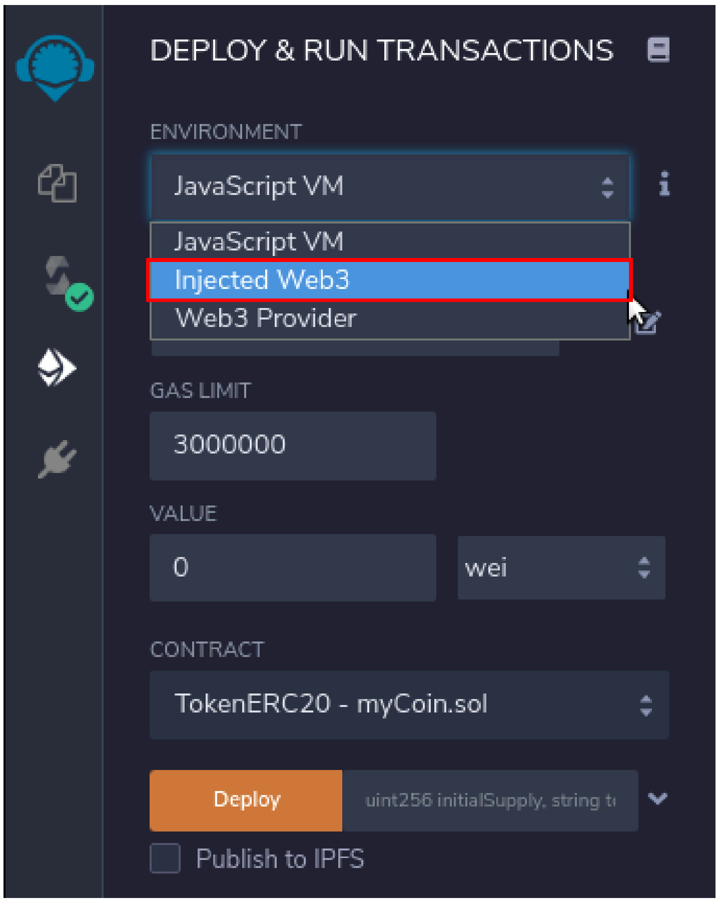
- After a few seconds, you may see that the faucet has transferred 1 Ether to your account



- Once you receive the Ether from the faucet you must deploy the contract on Ropsten.

  First, click on the **Deploy and Run Transactions** button and then select the **Injected Web3** option





DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: JavaScript VM

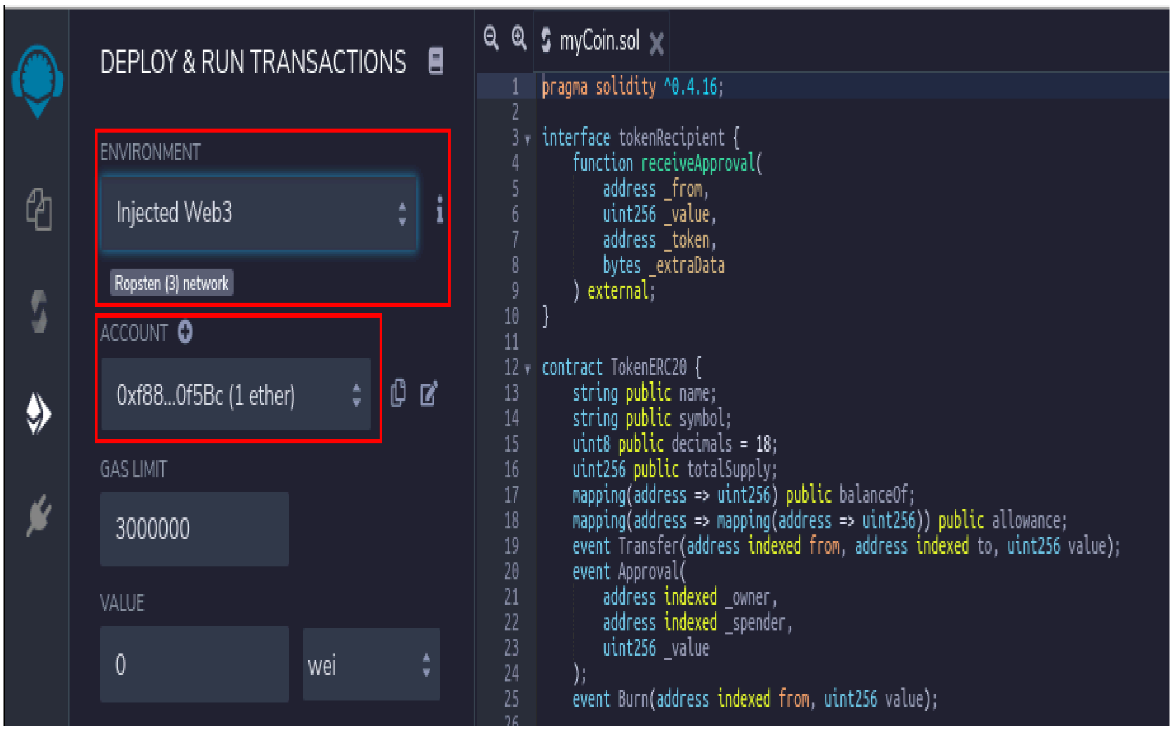
- JavaScript VM
- Injected Web3
- Web3 Provider

GAS LIMIT: 3000000

VALUE: 0 wei

CONTRACT: TokenERC20 - myCoin.sol

- The IDE connects to the Metamask account in the Ropsten network, as shown in the image below:



DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Injected Web3

Ropsten (3) network

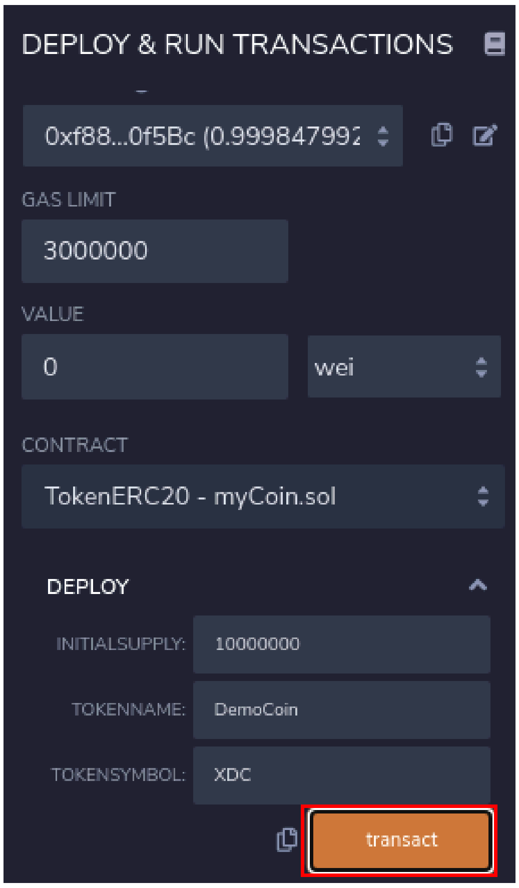
ACCOUNT: 0xf88...0f5Bc (1 ether)

GAS LIMIT: 3000000

VALUE: 0 wei

```
pragma solidity ^0.4.16;

interface tokenRecipient {
    function receiveApproval(
        address _from,
        uint256 _value,
        address _token,
        bytes _extraData
    ) external;
}

contract TokenERC20 {
    string public name;
    string public symbol;
    uint8 public decimals = 18;
    uint256 public totalSupply;
    mapping(address => uint256) public balanceOf;
    mapping(address => mapping(address => uint256)) public allowance;
    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(
        address indexed _owner,
        address indexed _spender,
        uint256 _value
    );
    event Burn(address indexed from, uint256 value);
```

- Set the value of **Initial Supply** as **10000000, Token Name** as **DemoCoin,** and the **Token Symbol** as **XDC** (or any other value you wish to use)before deploying the contract


DEPLOY & RUN TRANSACTIONS

0xf88...0f5Bc (0.999847992

GAS LIMIT: 3000000

VALUE: 0 wei

CONTRACT: TokenERC20 - myCoin.sol

DEPLOY

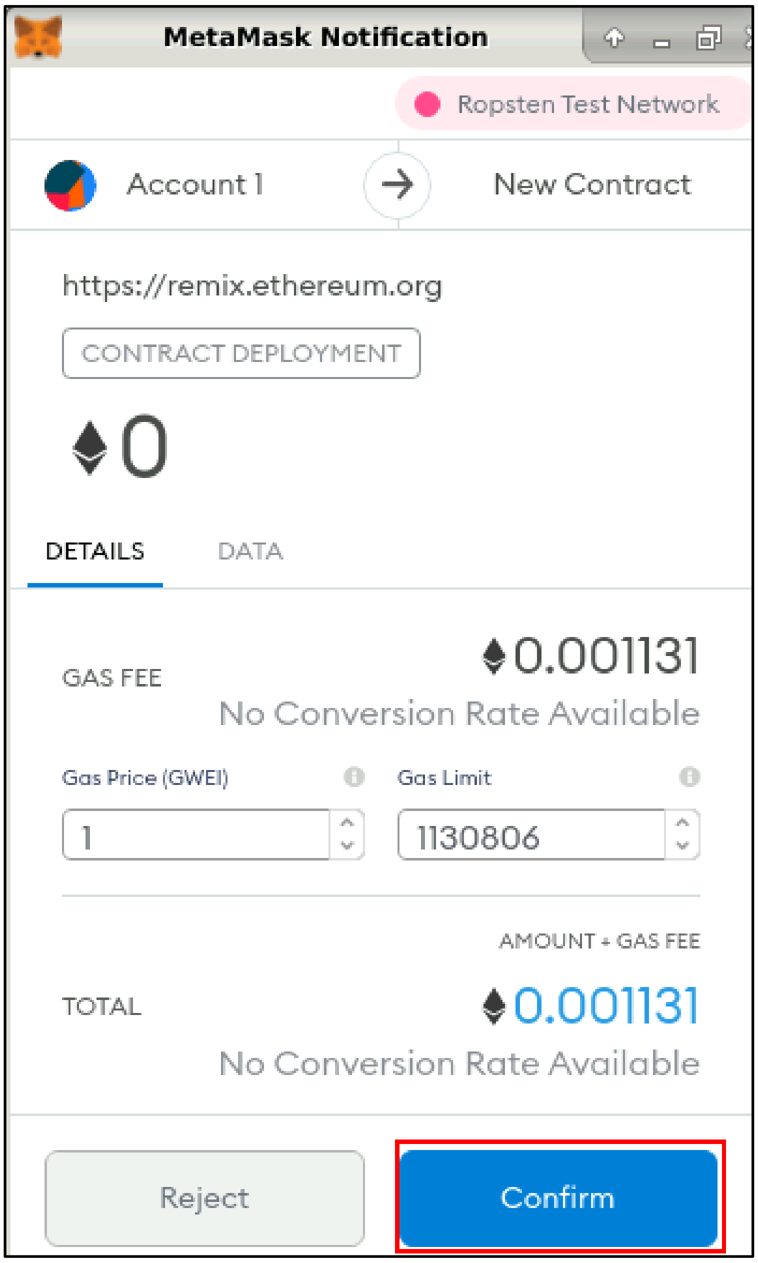
INITIALSUPPLY: 10000000

TOKENNAME: DemoCoin

TOKENSYMBOL: XDC

Step 3: Deploying the Contract on Ropsten Network using Metamask

- Once you click the **Transact** button in the previous step, it will prompt the Metamask dialog box as shown below, with the details of the transaction and the respective transaction fee to deploy the contract. Click on **Confirm** after all the details have been verified.


MetaMask Notification

Ropsten Test Network

Account 1 → New Contract

https://remix.ethereum.org

CONTRACT DEPLOYMENT

0

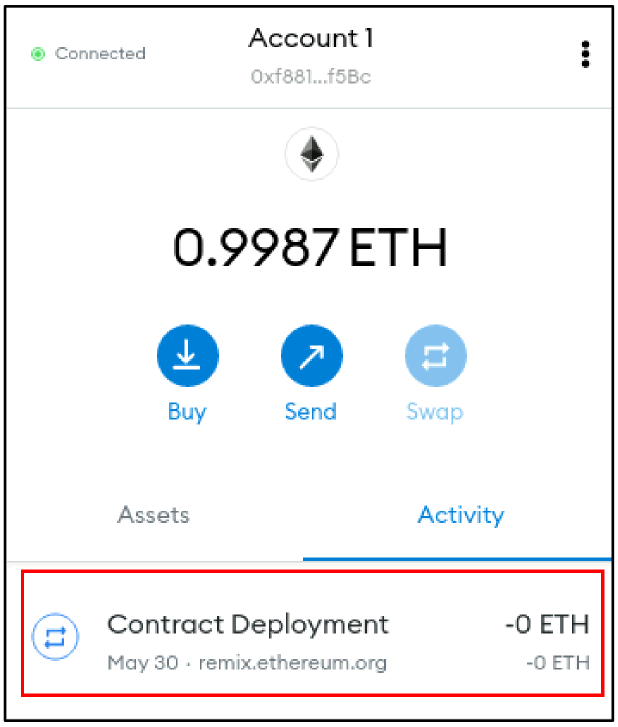
DETAILS DATA

GAS FEE 0.001131
No Conversion Rate Available

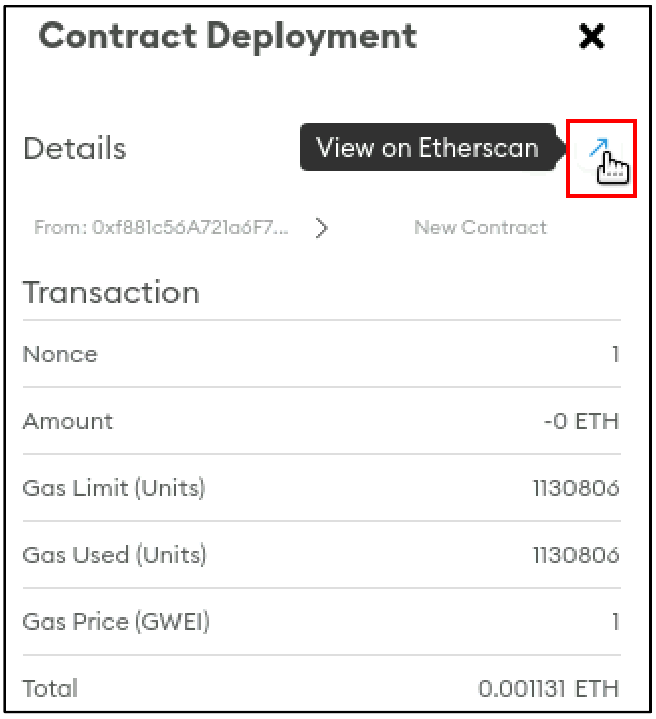
Gas Price (GWEI): 1
Gas Limit: 1130806

AMOUNT + GAS FEE
TOTAL 0.001131
No Conversion Rate Available

- Observe that the balance has been updated and the contract is reflected in the Activity section of Metamask. You can click on **Contract Deployment** to reveal its details



- You may view the complete details of deployment on Etherscan by clicking on the **View on Etherscan** arrow symbol


Contract Deployment

Details

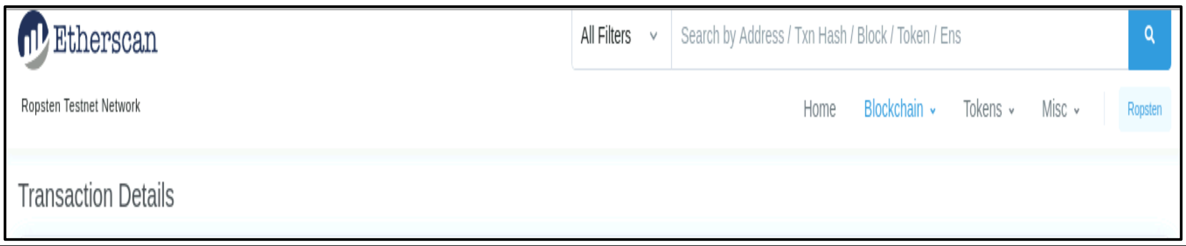
View on Etherscan

From: 0xf881c56A721a6F7... > New Contract

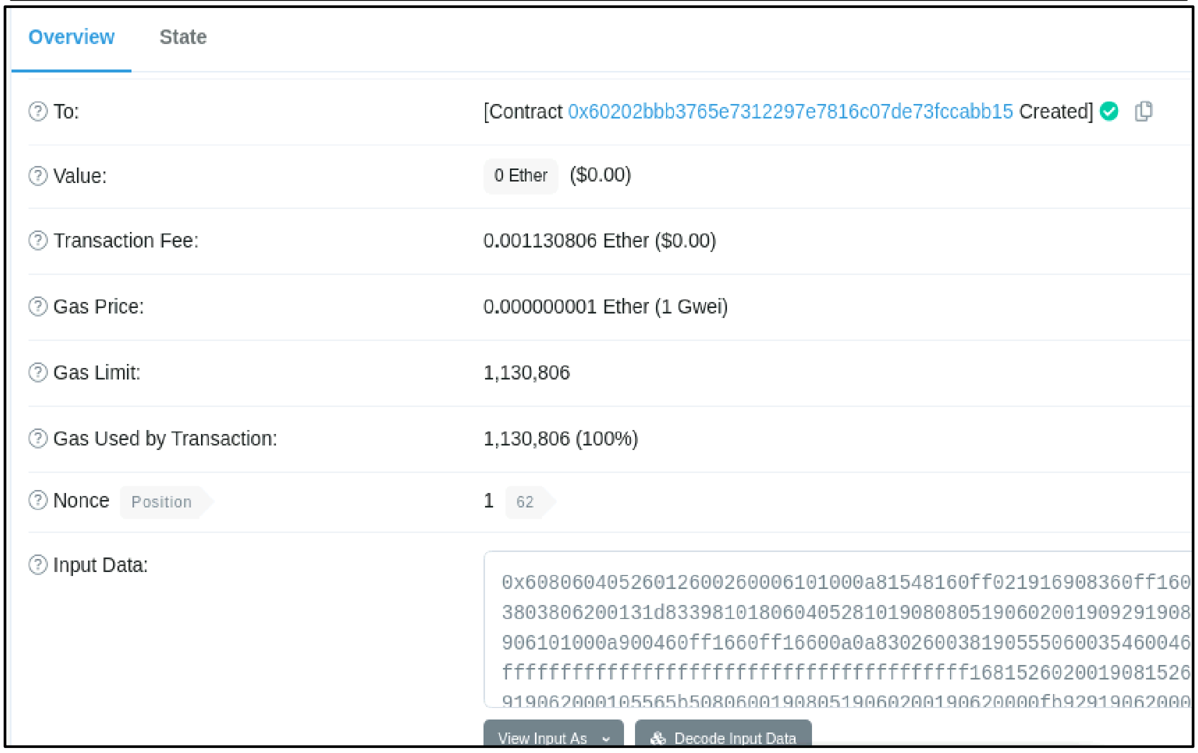
Transaction

Nonce	1
Amount	-0 ETH
Gas Limit (Units)	1130806
Gas Used (Units)	1130806
Gas Price (GWEI)	1
Total	0.001131 ETH

- Once on Etherscan, you can see the **Overview** of the contract details and its **State**

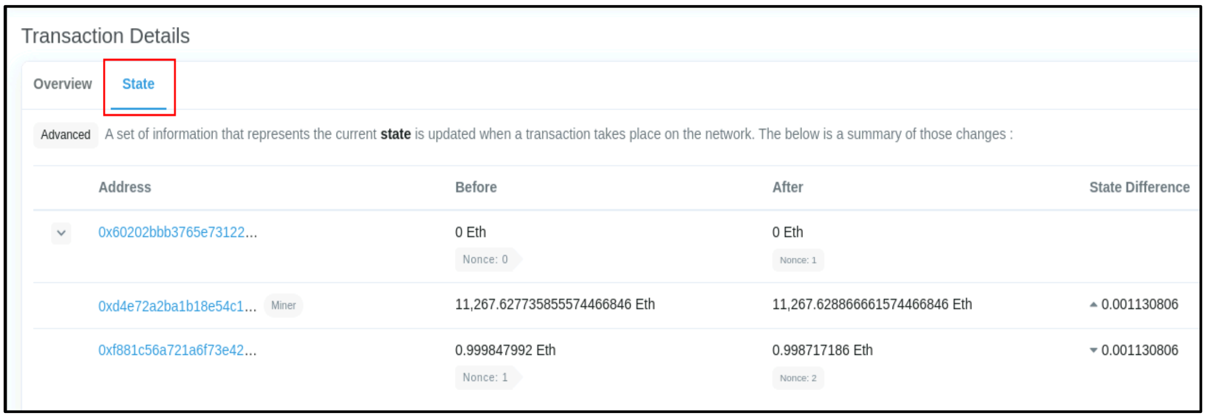


Transaction Details



To:	[Contract 0x60202bbb3765e7312297e7816c07de73fccabb15 Created]
Value:	0 Ether ($0.00)
Transaction Fee:	0.001130806 Ether ($0.00)
Gas Price:	0.000000001 Ether (1 Gwei)
Gas Limit:	1,130,806
Gas Used by Transaction:	1,130,806 (100%)
Nonce Position	1 62
Input Data:	

```
0x6080604052601260026000610100 0a81548160ff021916908360ff160
3803806200131d833981018060405281019080805190602001909291908
906101000a900460ff1660ff16600a0a83026003819055506003546004 6
ffffffffffffffffffffffffffffffffffffffff1681526020019081526
919062000105565b508060019080519060200190620000fb929190620000
```



Transaction Details

Advanced A set of information that represents the current **state** is updated when a transaction takes place on the network. The below is a summary of those changes :

Address	Before	After	State Difference
0x60202bbb3765e73122...	0 Eth Nonce: 0	0 Eth Nonce: 1	
0xd4e72a2ba1b18e54c1... Miner	11,267.627735855574466846 Eth	11,267.628866661574466846 Eth	▲ 0.001130806
0xf881c56a721a6f73e42...	0.999847992 Eth Nonce: 1	0.998717186 Eth Nonce: 2	▼ 0.001130806