

# Lab 8: Bitcoin and Blockchain

## 1 Details

---

Aim: To provide a foundation in understanding in Bitcoin and Blockchain.

## 2 Activities

---

L1.1 Using [blockchain.info](https://blockchain.info), find the details of the genesis block:

**Date created:**

**Reward:**

**Number of transactions:**

**Size of block:**

**Which account received the mining reward for the genesis block (last four digits):**

**How many USD does the original miner have in the account they used for the first genesis record:**

**When did the genesis block creator stop trading?**

L1.2 Using [blockchain.info](https://blockchain.info), determine the following

**Total bitcoins in circulation:**

**Most recent hash block (last four hex digits):**

**Block reward per block:**

**Difficulty:**

**Average time between blocks:**

**Market capitalisation (USD):**

**24 hr price (USD):**

**24hr transactions (USD):**

**Hash rate:**

**Last successful miner:**

**Maximum block size:**

**Balance for 1GbVUSW5WJmRCpaCJ4hanUny77oDaWW4to:**

**L1.3** Download and create the Python file defined on this page:

<https://asecuritysite.com/encryption/bit>

Now run the Python file, and compare the results in L.1.2.

**Total bitcoins in circulation:**

**Most recent hash block (last four hex digits):**

**Block reward per block:**

**Difficulty:**

**Average time between blocks:**

**Market capitalisation (USD):**

**24 hr price (USD):**

**24hr transactions (USD):**

**Hash rate:**

**Balance for 1GbVUSW5WJmRCpaCJ4hanUny77oDaWW4to:**

## **C      Setting up your Ethereum wallet on Sopolea**

Demo: <https://www.youtube.com/watch?v=xeaDE8wgVVQ>

The Sopolea network allows a user to test an Ethereum application and using free Ether. Initially, set up your MetaMask wallet. A document to outline how you set this up is [here](#). Once you have set it up, answer the following:

- What is your public ID (just define the first four hex values)?
- Find out someone else's public ID, and send them 0.001 Ether. If you are doing the lab on your own, send it to Bill (ID: 0xbB15B38e4ef6aF154b89A2E57E03Cd5cbD752233).

- Can you see the transaction on the Ethereum network? An example of a wallet is [here](#).
- Can you see your transaction on the Ethereum network for the person you send it to?
- What was the transaction fee for the transfer? If you were using the main Ethereum network, how much would the transaction cost in Dollars?
- Ask someone to send you 0.001 Ether. Did you receive it? If you are doing the lab on your own, ask your lab tutor to send you 0.001 Ether.

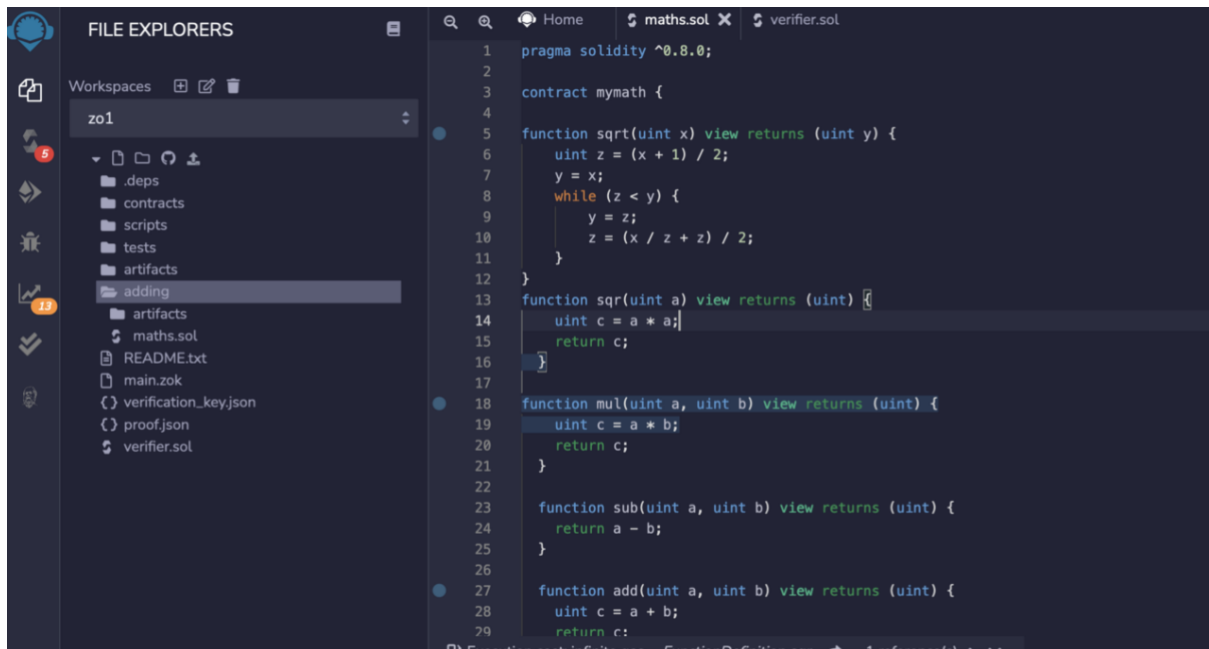
## Creating a Smart Contract in Ethereum

---

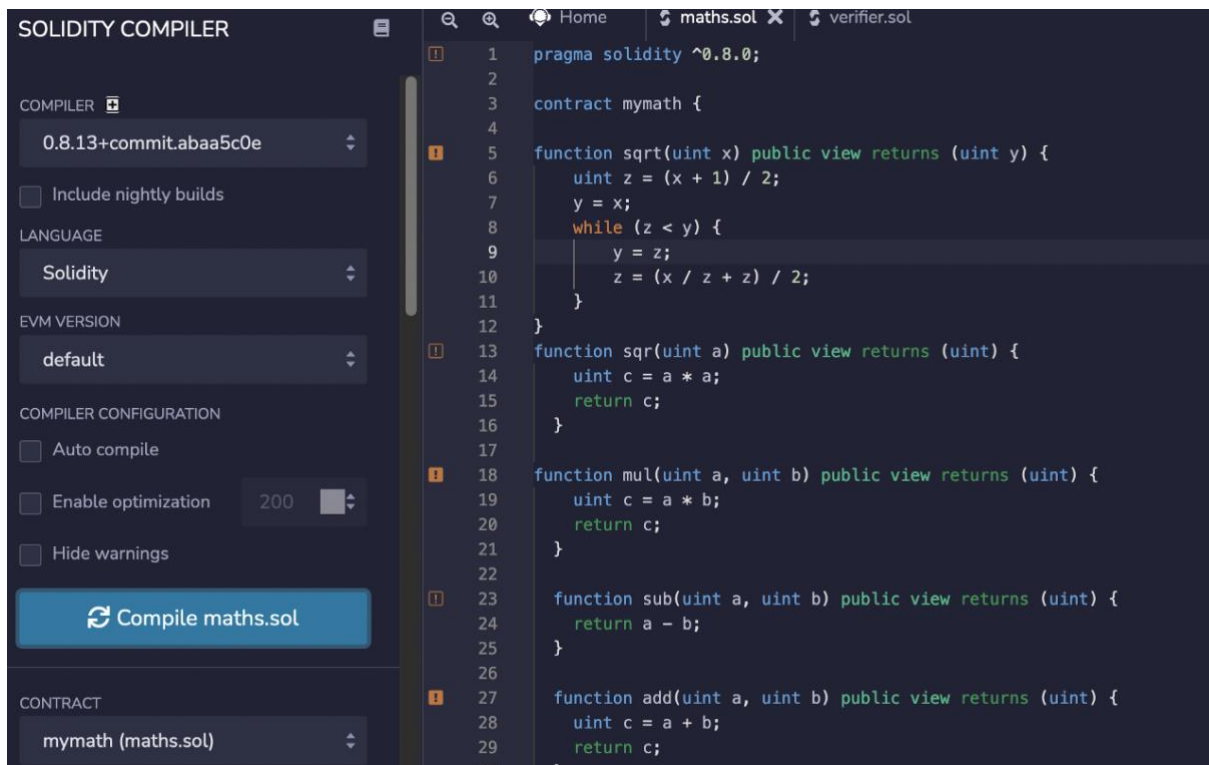
So, let's write a bit of code that does some simple maths. In the following we will implement `sqr()`, `sqr()`, `mul()`, `sub()`, and `add()`:

```
pragma solidity ^0.8.0;
contract mymath {function sqrt(uint x) public view returns (uint y) {
    uint z = (x + 1) / 2;
    y = x;
    while (z < y) {
        y = z;
        z = (x / z + z) / 2;
    }
}
function sqr(uint a) public view returns (uint) {
    uint c = a * a;
    return c;
}
function mul(uint a, uint b) public view returns (uint) {
    uint c = a * b;
    return c;
}
function sub(uint a, uint b) public view returns (uint) {
    return a - b;
}
function add(uint a, uint b) public view returns (uint) {
    uint c = a + b;
    return c;
}}
```

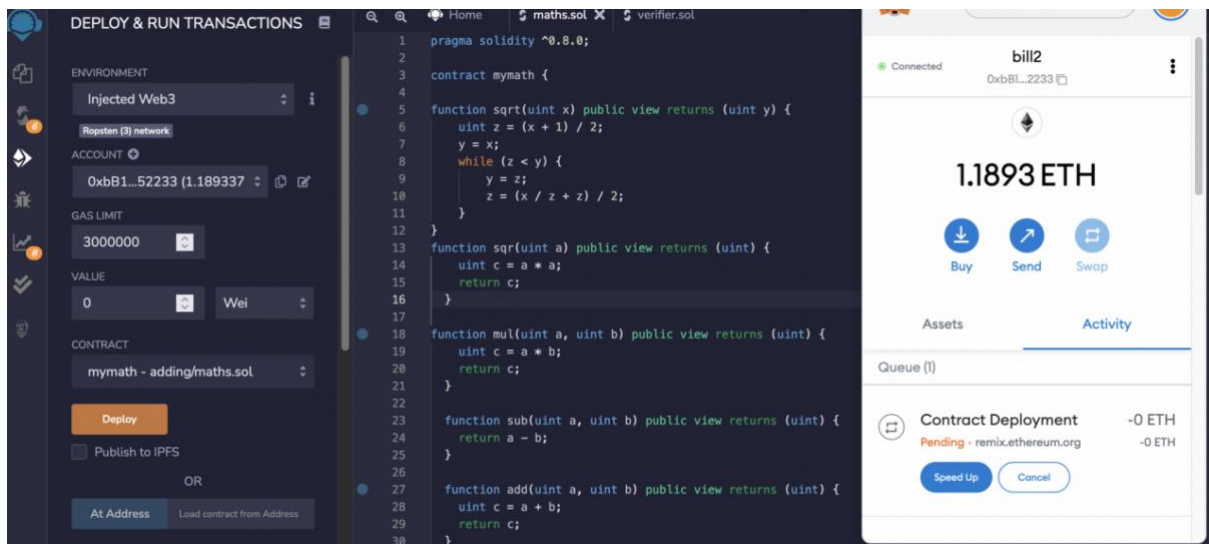
In this case, the "public" part makes sure we can see the output of the function, and the "view" part allows it to be stateless (and where we just have to receive the value without the smart contract remembering the state). On Ethereum we normally use the Solidity language to create a smart contract and then compile it into the byte code required for the ledger. First, can we start by entering the Solidity code into Remix [\[here\]](#):



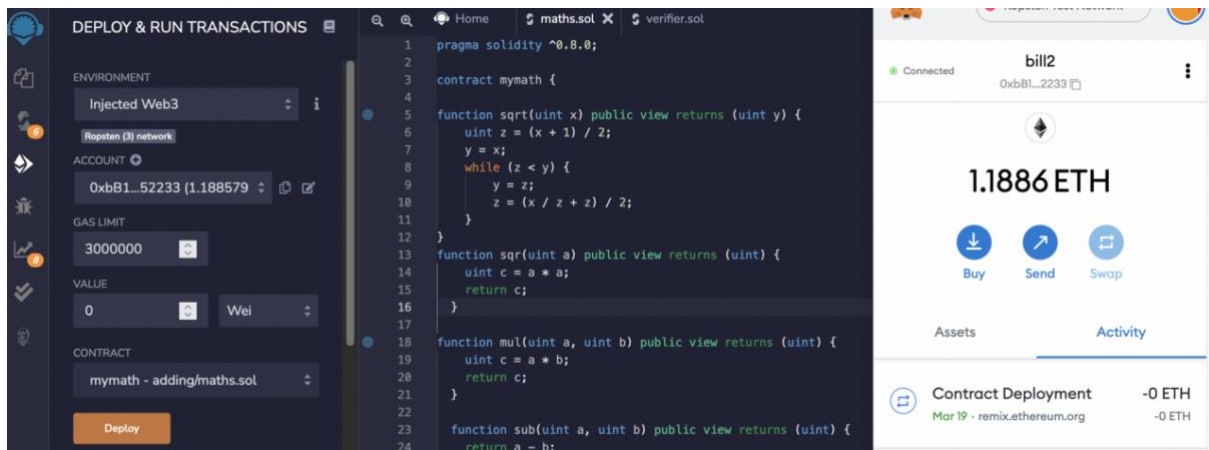
Once entered, we can then compile it with the Solidity compiler. It is important to take a note of the compiler version at this stage, as we will need this later:



Once compiled we can then deploy the smart contract to a test network (Ropsten). For this , we need to connect our MetaMask wallet:



Once it has been deployed, we can see our wallet identifies the deployed contract:



And clicking through gives us the address of the contract, and then viewing it on the explorer, we can see the transaction:

The screenshot shows the Etherscan interface for a transaction on the Ropsten Testnet. The transaction is confirmed and successful. A modal titled "Contract Deployment" is open, showing details for a new contract created by address 0xbb15b38e4ef6af154b89a2e57e03cd5cbd752233. The contract's nonce is 8, and it was deployed with 0 ETH. The transaction fee is 0.00075883 ETH.

Field	Value
Status	Confirmed
From	0xbb15b38e4ef6af154b89a2e57e03cd5cbd752233
To	New Contract
Transaction Nonce	8
Amount	-0 ETH
Gas Limit (Units)	303531
Gas Used (Units)	303531
Base Fee (GWEI)	0.000000008
Priority Fee (GWEI)	2.5
Total Gas Fee	0.000759 ETH
Max Fee Per Gas	0.000000003 ETH
Total	0.00075883 ETH


The address here is “0x0895..”, so we can view the smart contract from: here. We now need to verify and publish the contract, with click on “Verify and Publish”:

The screenshot shows the Etherscan interface for a specific contract. The contract is identified by the address 0x0895A540Cf8E7829284F1D9c55daF624D6e2Df9. The page includes a "Contract Overview" section showing a balance of 0 Ether. A "More Info" section indicates that the name tag is not available and the contract creator is 0xbb15b38e4ef6af154b8... at txn 0xce641d9aac54b17f30... The "Transactions" tab is selected, showing a list of transactions. A message prompts the user to verify and publish their contract source code today.

After this, we can define the Compiler Version and the licence

## Verify & Publish Contract Source Code

COMPILER TYPE AND VERSION SELECTION



Source code verification provides **transparency** for users interacting with smart contracts. By uploading the source code, Etherscan will match the compiled code with that on the blockchain. Just like contracts, a "smart contract" should provide end users with more information on what they are "digitally signing" for and give users an opportunity to audit the code to independently verify that it actually does what it is supposed to do.

Please enter the Contract Address you would like to verify

Please select Compiler Type

Solidity (Single file)
⌵

Please select Compiler Version

v0.8.13+commit.abaa5c0e
⌵

☒ Un-Check to show all nightly Commits also

Please select Open Source License Type [?](#)


2) The Unlicense (Unlicense)
⌵

☒ I agree to the [terms of service](#)

Continue

Reset

We then need to add your code for it to be checked:



**Etherscan**

Ropsten Testnet Network

All Filters

Search by Address / Txn Hash / Block / Token / Ens

Q

[Home](#)
[Blockchain](#)
[Tokens](#)
[Misc](#)
[Ropsten](#)

### Verify & Publish Contract Source Code

Compiler Type: SINGLE FILE / CONCATENATED METHOD

Info: A simple and structured interface for verifying smart contracts that fit in a single file

**Contract Source Code**

1. If the contract compiles correctly at [REMIX](#), it should also compile correctly here.

2. We have limited support for verifying contracts created by another contract and there is a timeout of up to 45 seconds for each contract compiled.

3. For programatic contract verification, check out the [Contract API Endpoint](#)

Contract Address

Compiler

v0.8.13+commit.abaa5c0e
⌵

[?](#) Optimization

No
⌵

**Enter the Solidity Contract Code below \***

```
pragma solidity ^0.8.0;

contract mymath {

function sqrt(uint x) public view returns (uint y) {
    uint z = (x + 1) / 2;
```

[Fetch from Gist](#)

It takes around 30 seconds, but, eventually, we should have our code accepted:

Etherscan  
Ropsten Testnet Network

All Filters Search by Address / Txn Hash / Block / Token / Ens

Home Blockchain Tokens Misc Ropsten

### Verify & Publish Contract Source Code

Compiler Type: SINGLE FILE / CONCATENATED METHOD

Info: A simple and structured interface for verifying smart contracts that fit in a single file

Contract Source Code **Compiler Output**

Compiler debug log:  
 Note: Contract was created during TxHash# 0xce641d9aac54b17f307dfe3a4c6bb2ae080e0dc7f1255d22297d83a250f75c0  
 Successfully generated ByteCode and ABI for Contract Address [0x0895a540cf8e7829284f1d9c55daf624d6e2df9]

Compiler Version: v0.8.13+commit.abaa5c0e  
 Optimization Enabled: 0  
 Runs: 200

ContractName:  
 mymath

ContractBytecode:  
 608060405234801561001057600080fd5b5061048a806100206000396000f3fe608060405234801561001057600080fd5b50600436106100575760003560e01c8063677342ce1461005c578063771602f71461008c578063a1a6c035146100bc578063b67d77c5146100ec578063c8a4ac9c1461011c575b600080fd5b6100766004803603810190610071919061024a565b61014c565b6040516100839190610286565b60405180910390f35b6100a660048036038101906100a191906102a1565b6101a9565b6040516100b39190610286565b60405180910390f35b6100d660048036038101906100d1919061024a565b6101c4565b6040516100e39190610286565b60405180910390f35b610106600480360381019061010191906102a1565b6101de565b6040516101139190610286565b60405180910390f3

We now have the contract published to the Ropsten test network:

Etherscan  
Ropsten Testnet Network

All Filters Search by Address / Txn Hash / Block / Token / Ens

Home Blockchain Tokens Misc Ropsten

**Contract** 0x0895A540Cf8E7829284F1D9c55daF624D6e2Df9

Contract Overview

Balance: 0 Ether

More Info

My Name Tag: Not Available

Contract Creator: 0xbb15b38e4ef6af154b8... at txn 0xce641d9aac54b17f30...

Transactions Contract Events

Latest 1 from a total of 1 transactions

Txn Hash	Method	Block	Age	From	To	Value	Txn Fee
0xce641d9aac54b17f30...	0x50806040	12106559	2 mins ago	0xbb15b38e4ef6af154b8...	IN Contract Creation	0 Ether	0.000758827502

[ Download CSV Export ]

Next, by selected the Contract tab, and can view the read parameters. The exposed functions are add(), mul(), sqr(), sqrt() and sub():



To test, we can just enter the variables for a given function, and get a result:

## Creating ERC-20 tokens

Within the Ethereum blockchain, we can record transactions and run smart contracts. These things allow us to run DApps (decentralized applications) and which can support the running of the infrastructure in return for some payment (Ether). A DApp can also create tokens for new currencies, shares in a company or to prove the ownership of an asset. ERC-20 is a standard format for a Fungible Token and which can support the sharing, transfer and storage of tokens. These tokens are supported by the whole of the Ethereum infrastructure and can be easily traded. They support a number of mandatory functions:

- `totalSupply`. This function is the total number of ERC-20 tokens that have been created.
- `balanceOf`. This function identifies the number of tokens that a given address has in its account.
- `transfer`. This function supports the transfer of tokens to a defined user address.
- `transferFrom`. This function supports a user to transfer tokens to another user.
- `approve`. This function checks that a transaction is valid, based on the supply of token.
- `allowance`. This function checks if a user has enough funds in their account for a transaction.

There are also a number of options:

- `Token Name`. This is the name that the token will be defined as.
- `Symbol`. This is the symbol that the token will use.
- `Decimal`. This is the number of decimal places to be used for any transactions.

Now we you create your own token. If you are Bob Smith, then call your token "BobSmithToken", and your currency will be "BobSmith".

So, let's create a token named "ENUToken" (change the name to your name), and use the tutorial sample from here. First, we open up <https://remix.ethereum.org/>, and enter the following Solidity contract:

```
pragma solidity ^0.4.24;

// -----
// 'ENU Token' token contract
//
// Deployed to : 0xbB15B38e4ef6aF154b89A2E57E03Cd5cbD752233
// Symbol      : ENUToken
// Name        : ENU Token
// Total supply: 100000000
// Decimals    : 18

// Based on https://github.com/bitfwdcommunity/Issue-your-own-ERC20-
// token/tree/master/contracts

// -----
// 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;
    }
    function safeMul(uint a, uint b) public pure returns (uint c) {
```

```

        c = a * b;
        require(a == 0 || c / a == b);
    }
    function safeDiv(uint a, uint b) public pure returns (uint c) {
        require(b > 0);
        c = a / b;
    }
}

// -----
// ERC Token Standard #20 Interface
// https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-
// standard.md
// -----
contract ERC20Interface {
    function totalSupply() public constant returns (uint);
    function balanceOf(address tokenOwner) public constant returns (uint
balance);
    function allowance(address tokenOwner, address spender) public constant
returns (uint remaining);
    function transfer(address to, uint tokens) public returns (bool
success);
    function approve(address spender, uint tokens) public returns (bool
success);
    function transferFrom(address from, address to, uint tokens) public
returns (bool success);

    event Transfer(address indexed from, address indexed to, uint tokens);
    event Approval(address indexed tokenOwner, address indexed spender,
uint tokens);
}

// -----
// Contract function to receive approval and execute function in one call
//
// Borrowed from MiniMeToken
// -----
contract ApproveAndCallFallBack {
    function receiveApproval(address from, uint256 tokens, address token,
bytes data) public;
}

// -----
// Owned contract
// -----
contract Owned {
    address public owner;
    address public newOwner;

    event OwnershipTransferred(address indexed _from, address indexed _to);

    constructor() public {

```

[illegible]

```

        return _totalSupply - balances[address(0)];
    }

    // -----
    // Get the token balance for account tokenOwner
    // -----
    function balanceOf(address tokenOwner) public constant returns (uint
balance) {
        return balances[tokenOwner];
    }

    // -----
    // Transfer the balance from token owner's account to to account
    // - Owner's account must have sufficient balance to transfer
    // - 0 value transfers are allowed
    // -----
    function transfer(address to, uint tokens) public returns (bool
success) {
        balances[msg.sender] = safeSub(balances[msg.sender], tokens);
        balances[to] = safeAdd(balances[to], tokens);
        emit Transfer(msg.sender, to, tokens);
        return true;
    }

    // -----
    // Token owner can approve for spender to transferFrom(...) tokens
    // from the token owner's account
    //
    // https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-
standard.md
    // recommends that there are no checks for the approval double-spend
attack
    // as this should be implemented in user interfaces
    // -----
    function approve(address spender, uint tokens) public returns (bool
success) {
        allowed[msg.sender][spender] = tokens;
        emit Approval(msg.sender, spender, tokens);
        return true;
    }

    // -----
    // Transfer tokens from the from account to the to account
    //
    // The calling account must already have sufficient tokens
approve(...) -d
    // for spending from the from account and
    // - From account must have sufficient balance to transfer
    // - Spender must have sufficient allowance to transfer
    // - 0 value transfers are allowed

```

```

// -----
----
function transferFrom(address from, address to, uint tokens) public
returns (bool success) {
    balances[from] = safeSub(balances[from], tokens);
    allowed[from][msg.sender] = safeSub(allowed[from][msg.sender],
tokens);
    balances[to] = safeAdd(balances[to], tokens);
    emit Transfer(from, to, tokens);
    return true;
}

// -----
----
// Returns the amount of tokens approved by the owner that can be
// transferred to the spender's account
// -----
----
function allowance(address tokenOwner, address spender) public constant
returns (uint remaining) {
    return allowed[tokenOwner][spender];
}

// -----
----
// Token owner can approve for spender to transferFrom(...) tokens
// from the token owner's account. The spender contract function
// receiveApproval(...) is then executed
// -----
----
function approveAndCall(address spender, uint tokens, bytes data)
public returns (bool success) {
    allowed[msg.sender][spender] = tokens;
    emit Approval(msg.sender, spender, tokens);
    ApproveAndCallFallBack(spender).receiveApproval(msg.sender, tokens,
this, data);
    return true;
}

// -----
----
// Don't accept ETH
// -----
----
function () public payable {
    revert();
}

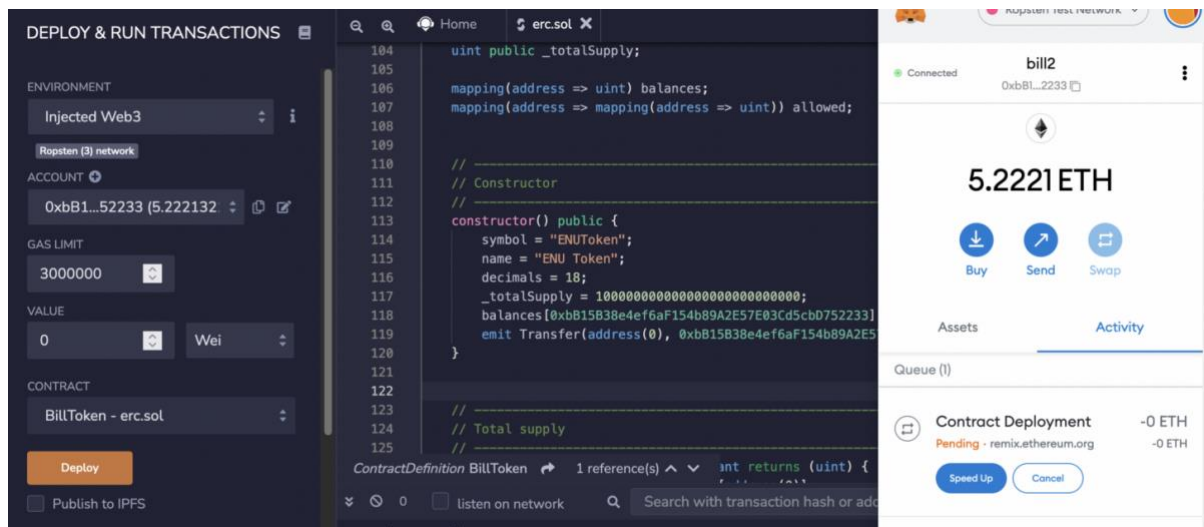
// -----
----
// Owner can transfer out any accidentally sent ERC20 tokens
// -----
----
function transferAnyERC20Token(address tokenAddress, uint tokens)
public onlyOwner returns (bool success) {
    return ERC20Interface(tokenAddress).transfer(owner, tokens);
}
}

```

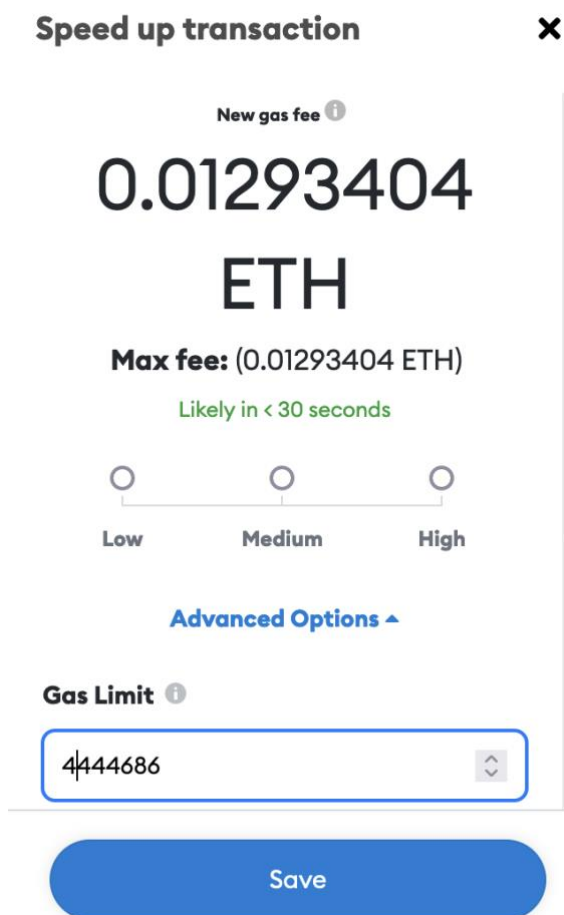
```
constructor() public {  
    symbol = "ENUToken";  
    name = "ENU Token";  
    decimals = 18;  
    _totalSupply = 10000000000000000000000000000;_  
    balances[0xbB15B38e4ef6aF154b89A2E57E03Cd5cbD752233] = _totalSupply;_  
    emit Transfer(address(0), 0xbB15B38e4ef6aF154b89A2E57E03Cd5cbD752233,  
_totalSupply);_  
}
```

[illegible]

After this, our contract will be shown as being pending deployment:

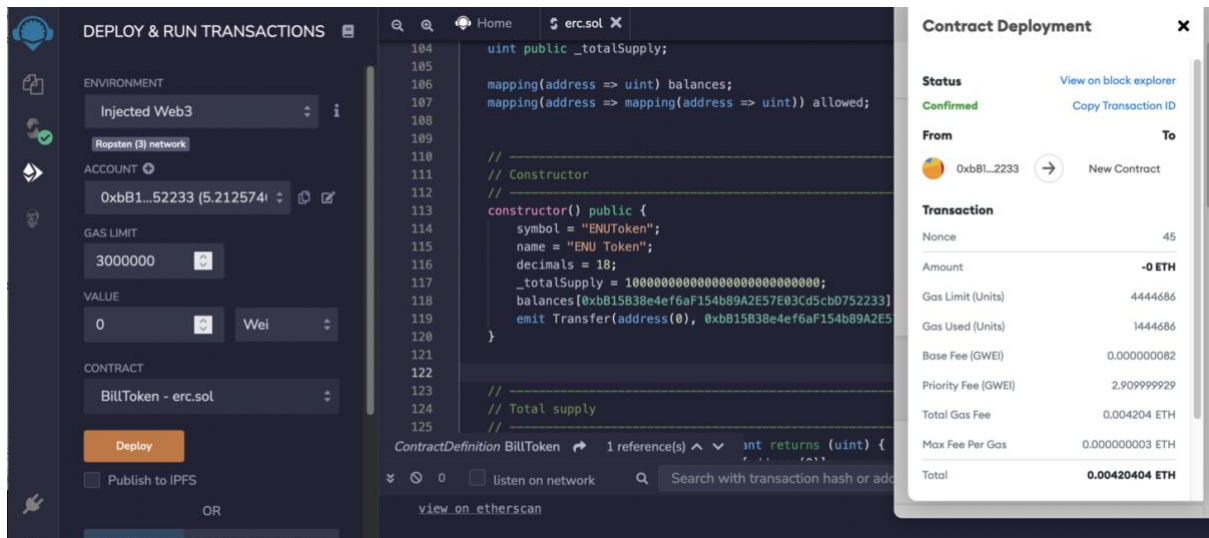


It will take 10–15 minutes to deploy, but it can be speeded up by increasing the gas limit:

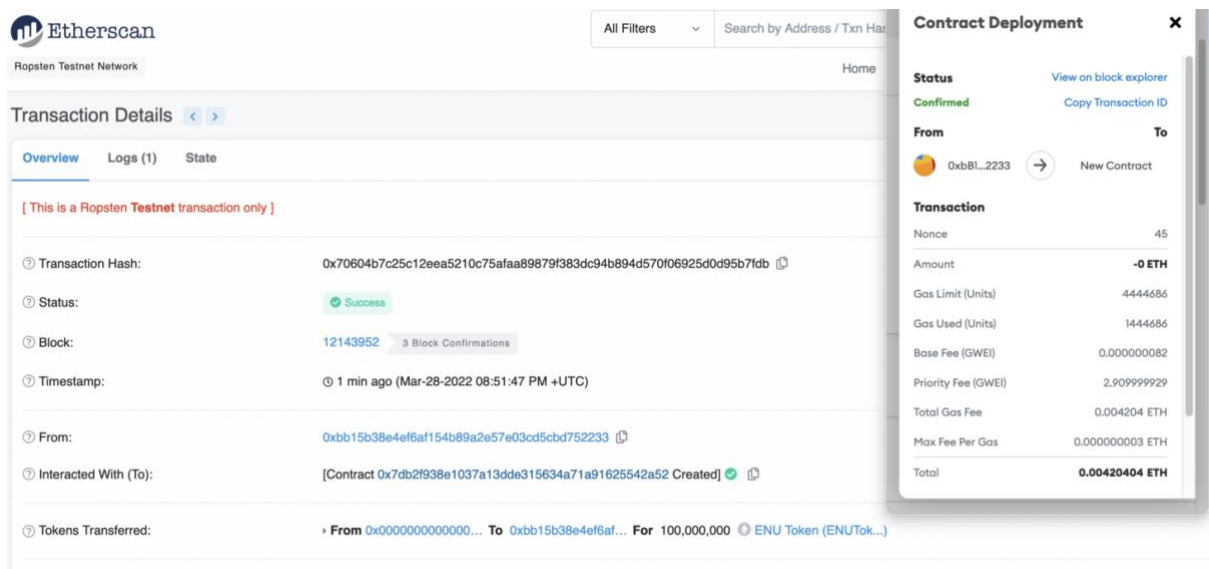


Once deployed, we can view the contract details:






And can then view the transaction for the contract [here]:




And then view the contract here]:


Etherscan

All Filters

Ropsten Testnet Network
Home
Blockchain
Tokens
Misc
Ropsten


Contract
0x7db2f938e1037A13dde315634A71a91625542a52

Contract Overview

Balance:

0 Ether

More Info


My Name Tag:

Not Available

Contract Creator:





0xbb15b38e4ef6af154b8... at txn 0x70604b7c25c12eea52...

Token Tracker:


ENU Token (ENUToken)

Transactions
Contract
Events

Latest 1 from a total of 1 transactions

Txn Hash	Method	Block	Age	From	To	Value	Txn Fee	
 <span>0x70604b7c25c12eea52...</span>	 <span>0x60806040</span>	<span>12143952</span>	<span>2 mins ago</span>	<span>0xbb15b38e4ef6af154b8...</span>	<span>IN</span>	 <span>Contract Creation</span>	<span>0 Ether</span>	<span>0.004204036275</span> 

[Download CSV Export]

[illegible]



[illegible]

**Address** 0xb15B38e4ef6aF154b89A2E57E03Cd5cbD752233
🔍 ⚙️

---

**Overview**

Balance: 5.212574667682053407 Ether

Token: \$0.00 ?

**More Info**

My Name Tag: Not Available

---

**Transactions**    Erc20 Token Txns

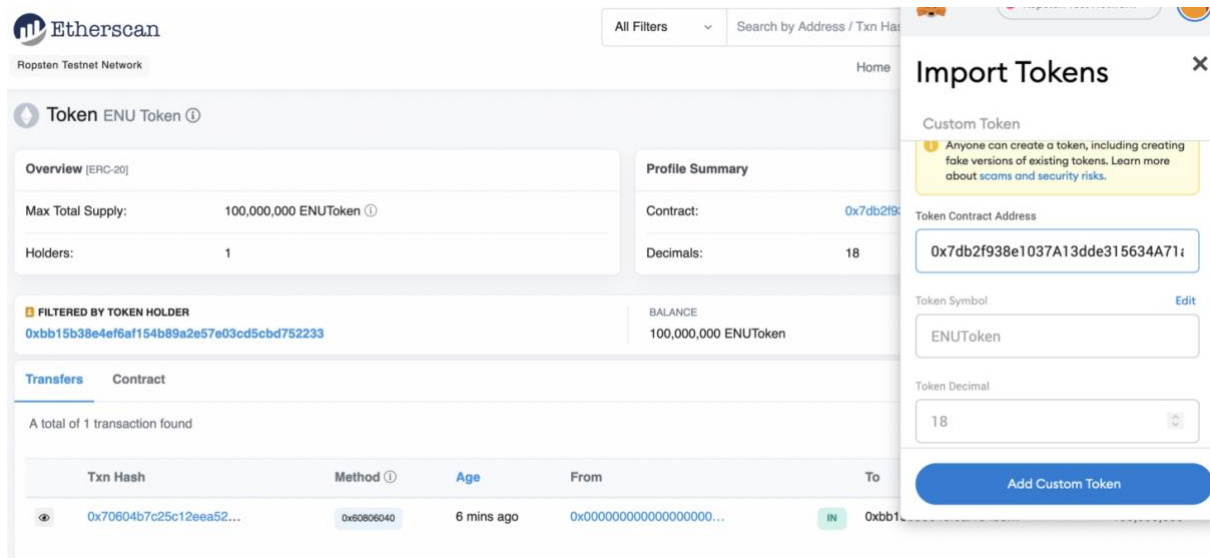
⌵ Latest 25 from a total of 60 trans

Txn Hash	From	To	Value	Txn Fee
0x70604b7c25c12eea52...	0xb15b38e4ef6af154b8...	Create: BillToken	0 Ether	0.004204036275...
0x388e659d2d3573594...	0x60806040	Create: BillToken	0 Ether	0.005290447133...
0xc582900527be214c91...	Accept Ownership	0x05a979ab299ca1d907...	0 Ether	0.000062979456...
0xcbb354a71b114733b7...	0x60806040	Create: Owned	0 Ether	0.001109332010...

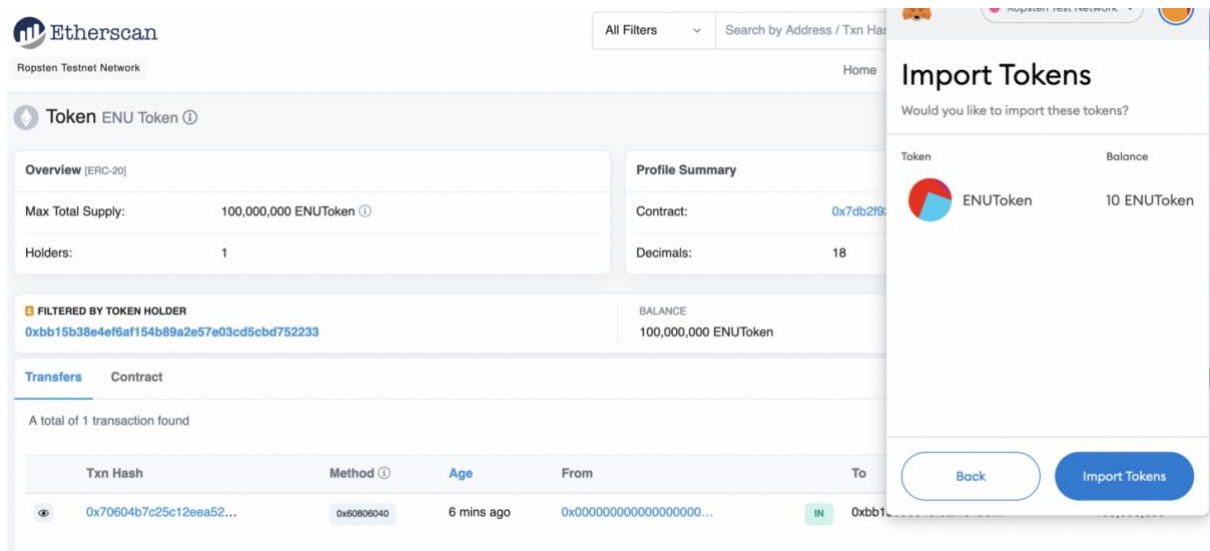
Search for Token Name

> ERC-20 Tokens (2)

- BillToken (BillTok...) 100,000,000 BillToken
- ENUToken (ENUTok...) 100,000,000 ENUToken



We will now have our new tokens in the wallet:



And with:

The screenshot shows the Etherscan interface for the ENUToken on the Ropsten Testnet. The token details include a Max Total Supply of 100,000,000 ENUToken and 1 holder. A transaction table shows a transfer from 0x00 to 0xbb15b38e4ef6af154b89a2e57e03cd5cbd752233. A mobile wallet overlay is visible on the right, showing a balance of 1000000... ENUToken and options to Send or Swap.

We can now transfer the cryptocurrency to another wallet:

The screenshot shows the Etherscan interface with the 'Send Tokens' modal open. The modal displays the Asset as ENUToken with a balance of 100000000 ENUToken. The Amount field is set to 0.01 ENUToken. The modal includes 'Cancel' and 'Next' buttons.

We can view the ENUToken: here]:

The screenshot shows the Etherscan interface with the 'Holders' tab selected for the ENUToken. It displays a table with 1 holder, showing the Txn Hash, Method, Age, From, To, and Quantity. The holder's address is 0xbb15b38e4ef6af154b89a2e57e03cd5cbd752233, and the quantity is 100,000,000 ENUToken.

Now answer the following:

- Do you see the tokens in your wallet?
- Now send 0.1 of your token to someone else's wallet. If you want, you can send to your tutor's wallet. Bill's wallet is 0xbb15b38e4ef6af154b89a2e57e03cd5cbd752233
- Did they receive the token?