

Creating a Crypto Token Using a Smart Contract

Campbell R. Harvey
Duke University and NBER

Developed in collaboration with Hyung-Yoon Kim, Anibal Granada Gonzales, Manmit Singh and Yash Patil

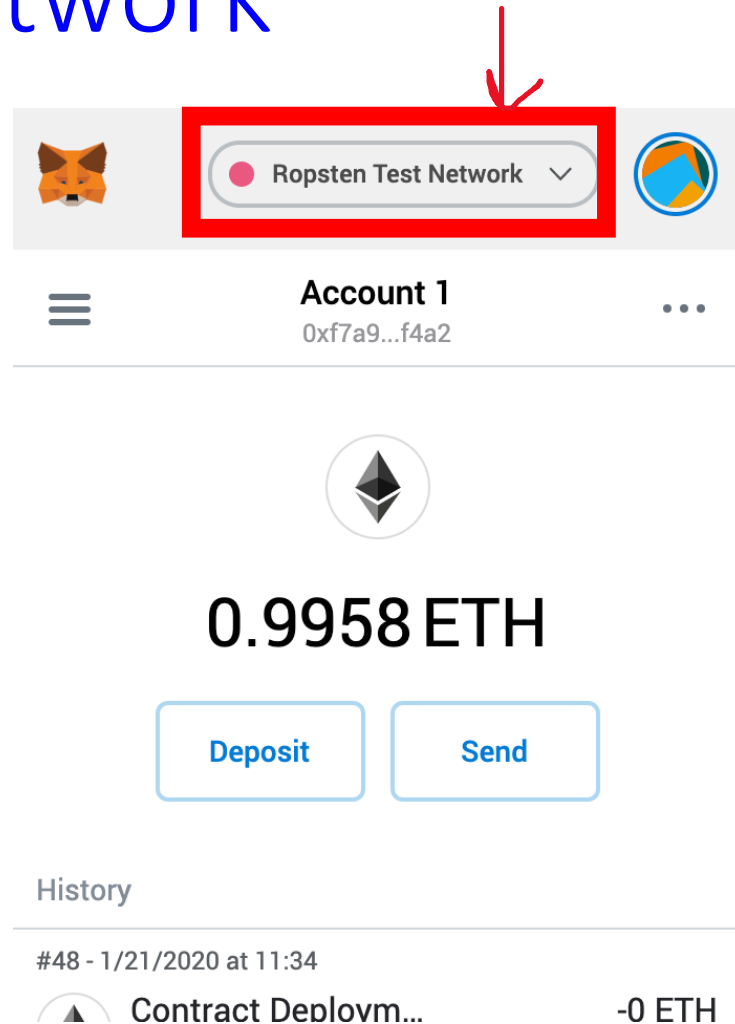
Things you'll need

- A basic understanding of the Solidity language
<https://cryptozombies.io/en/course/>
- Metamask extension: chrome web store
- Remix IDE: <https://remix.ethereum.org/>
- At least 1 full testnet ether: <https://faucet.ropsten.be/>

Parameters for the token

- Name (e.g., FQ1)
- Symbol (e.g., FQ1, default is SYM)
- Decimal places (e.g., 2 decimal places)
- Number of units in circulation (e.g., 100 billion)
- **The number of whole tokens will be 1 billion in the above example, i.e.,**
100 billion/100 (10² from 2 decimal places)

First off, make sure your MetaMask is on Ropsten Test Network

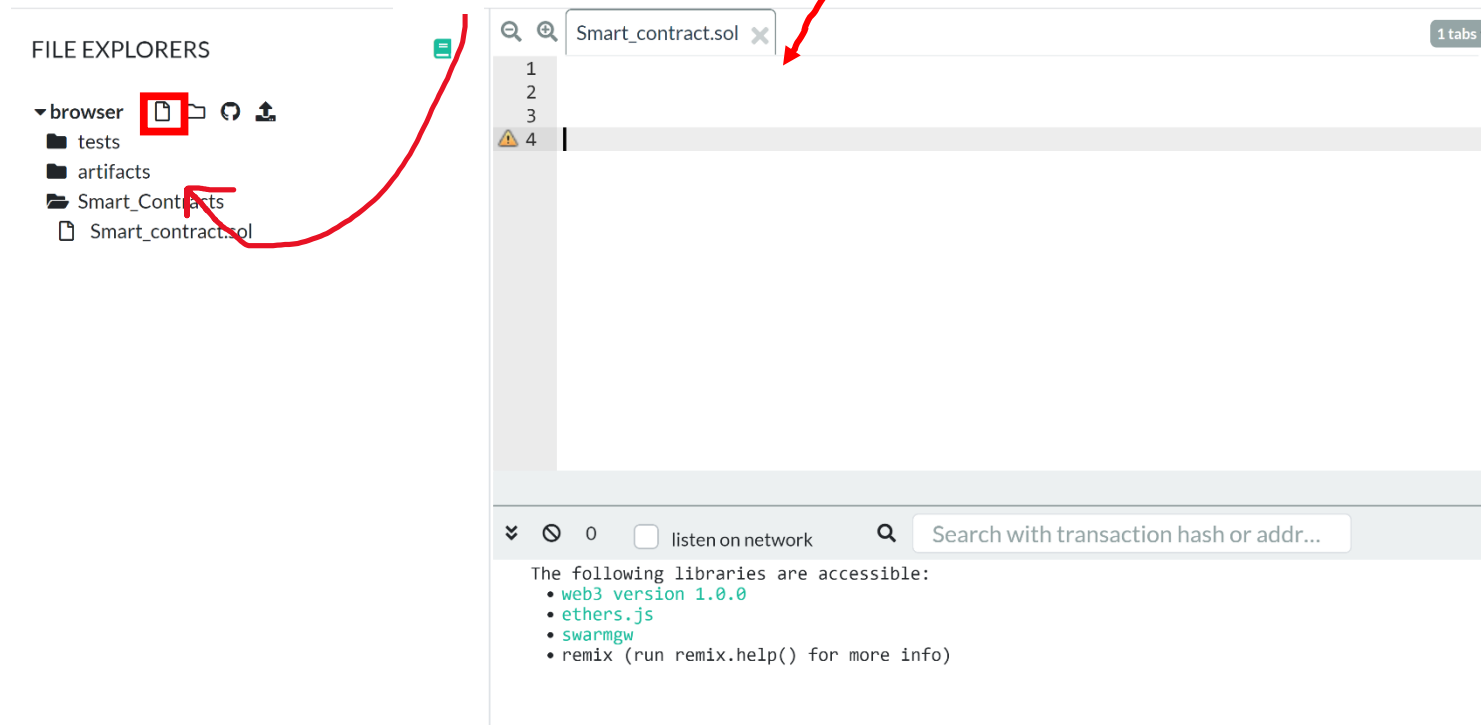


Step 1. Acquire test ethereum for the gas fee

Go to <https://faucet.metamask.io/> and request at least 1 Ethereum from the faucet

Step 2. Write a contract

1. Go to the IDE (Integrated Development Environment) <https://remix.ethereum.org>
2. Select the Solidity Environment
3. Create new file by clicking on 



Step 2. Write a contract

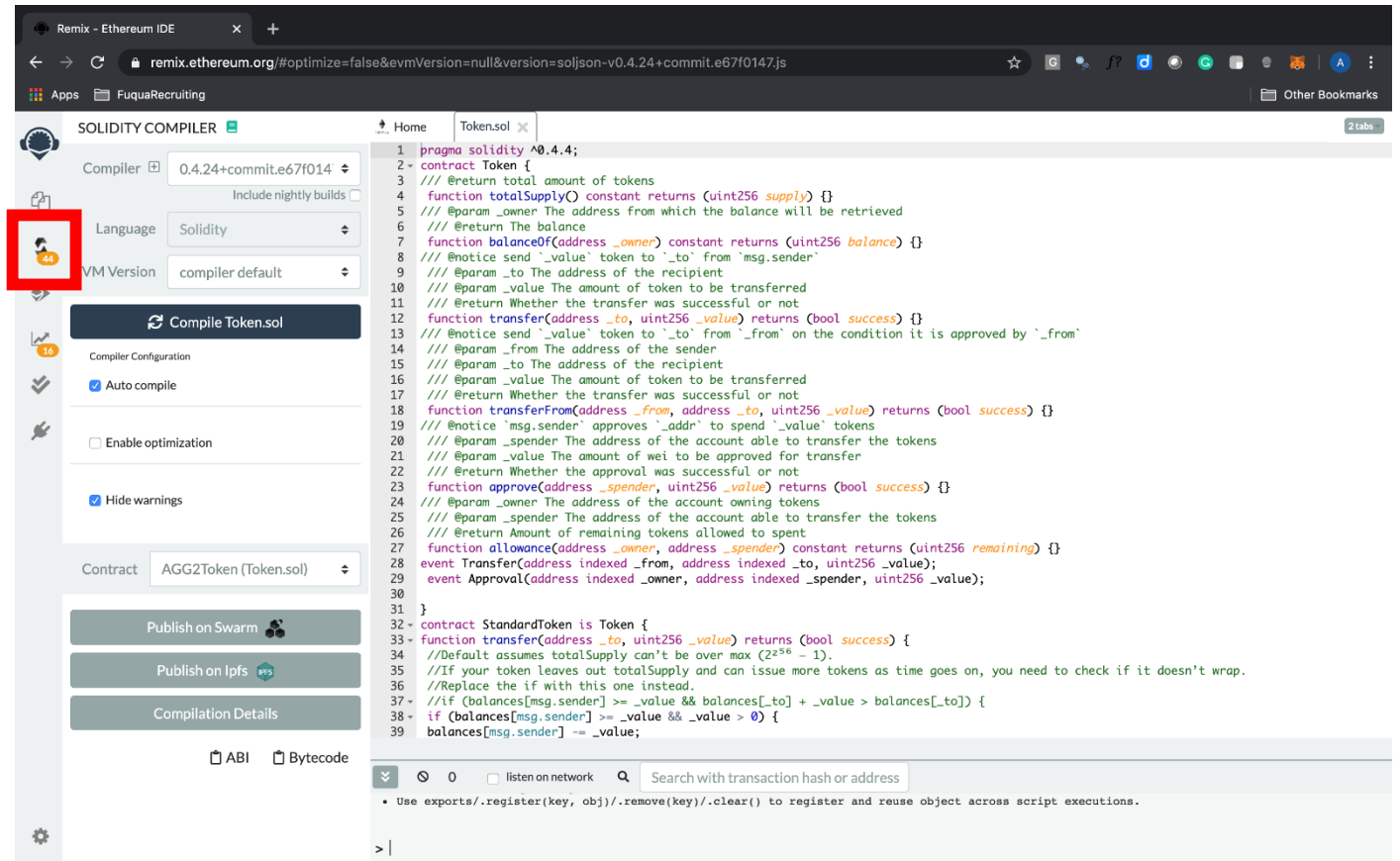
3. Name the file “Token.sol”

4. Copy and paste the contract code*

- Note: Be careful here because some copy and pastes might change some characters like ‘ or -.
- Depending on your browser, the code will be highlighted; noticed that all the code preceded by a slash (‘/’) takes the same color. **Solidity** understands text after / as comments and will not attempt to run it as code (similar to ‘#’ in **R** and **Python**).
- Notice that functions, object names and parameters take distinctive colors.

Step 3. Compile the contract

- 1. Switch to Compiler tab
- 2. Choose 0.4.24 commit version compiler
- 3. 'Auto compile' ON, 'Enable optimization' OFF, 'Hide warnings' ON
- 4. Compile the contract



The screenshot shows the Remix Ethereum IDE interface. The browser address bar displays `remix.ethereum.org/#optimize=false&evmVersion=null&version=soljson-v0.4.24+commit.e67f0147.js`. The main window is titled "SOLIDITY COMPILER" and shows the following configuration:

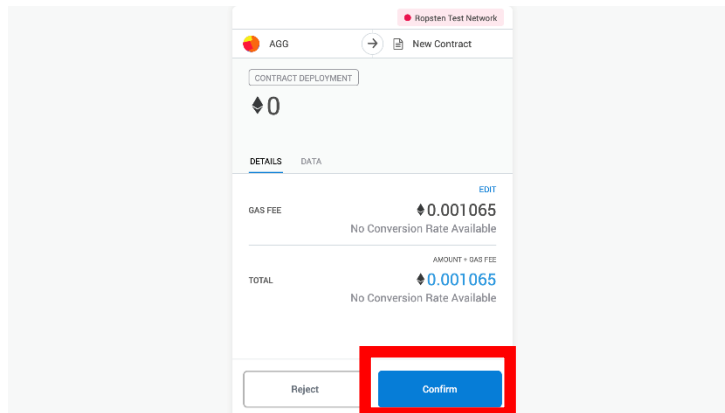
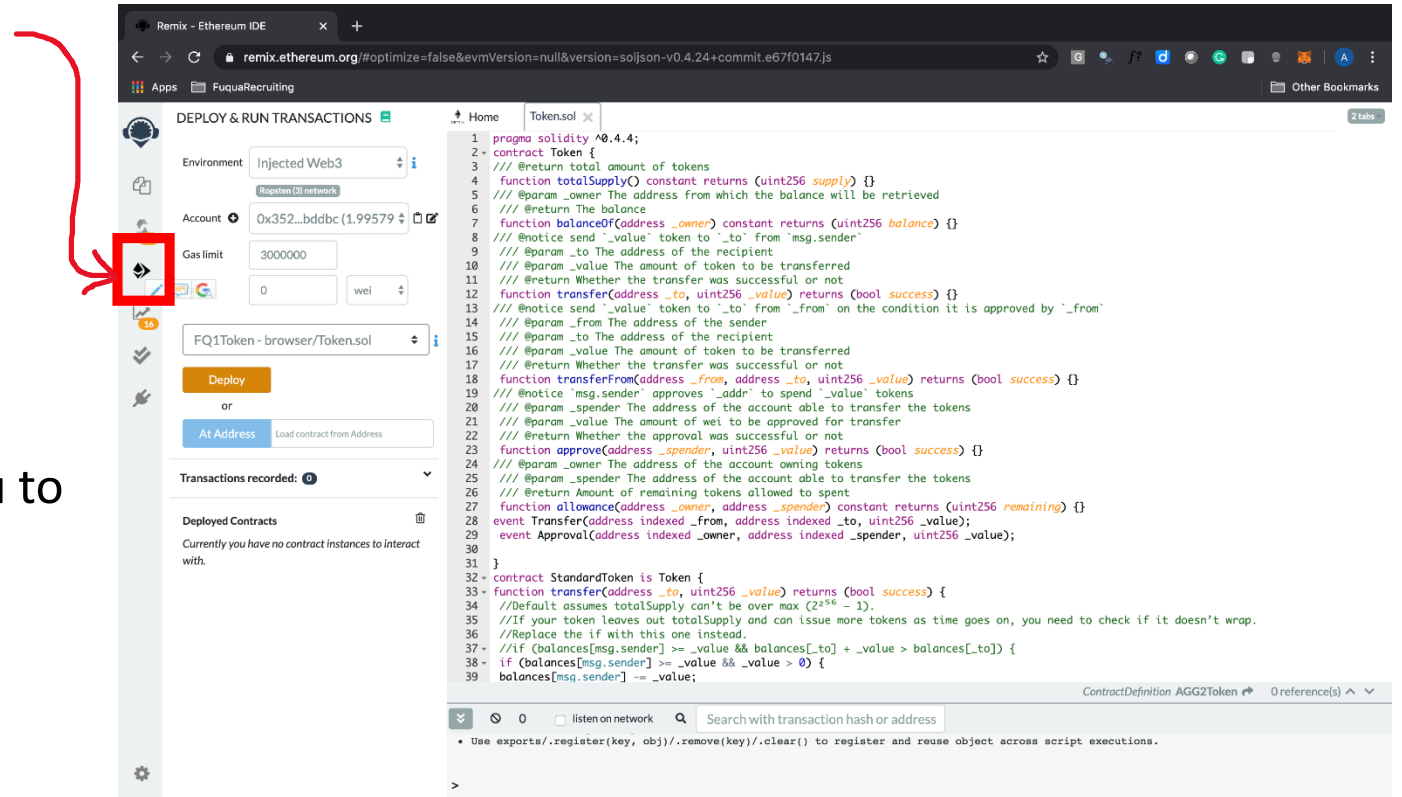
- Compiler: 0.4.24+commit.e67f014
- Language: Solidity
- VM Version: compiler default
- Buttons: Compile Token.sol, Publish on Swarm, Publish on Ipfs, Compilation Details
- Compiler Configuration:
 - Auto compile:
 - Enable optimization:
 - Hide warnings:
- Contract: AGG2Token (Token.sol)
- ABI and Bytecode options are visible at the bottom.

The Solidity code in the editor is as follows:

```
1 pragma solidity ^0.4.4;
2 contract Token {
3     /// @return total amount of tokens
4     function totalSupply() constant returns (uint256 suppl) {}
5     /// @param _owner The address from which the balance will be retrieved
6     /// @return The balance
7     function balanceOf(address _owner) constant returns (uint256 balance) {}
8     /// @notice send `_value` token to `_to` from `msg.sender`
9     /// @param _to The address of the recipient
10    /// @param _value The amount of token to be transferred
11    /// @return Whether the transfer was successful or not
12    function transfer(address _to, uint256 _value) returns (bool success) {}
13    /// @notice send `_value` token to `_to` from `_from` on the condition it is approved by `_from`
14    /// @param _from The address of the sender
15    /// @param _to The address of the recipient
16    /// @param _value The amount of token to be transferred
17    /// @return Whether the transfer was successful or not
18    function transferFrom(address _from, address _to, uint256 _value) returns (bool success) {}
19    /// @notice `msg.sender` approves `_addr` to spend `_value` tokens
20    /// @param _spender The address of the account able to transfer the tokens
21    /// @param _value The amount of wei to be approved for transfer
22    /// @return Whether the approval was successful or not
23    function approve(address _spender, uint256 _value) returns (bool success) {}
24    /// @param _owner The address of the account owning tokens
25    /// @param _spender The address of the account able to transfer the tokens
26    /// @return Amount of remaining tokens allowed to spent
27    function allowance(address _owner, address _spender) constant returns (uint256 remaining) {}
28    event Transfer(address indexed _from, address indexed _to, uint256 _value);
29    event Approval(address indexed _owner, address indexed _spender, uint256 _value);
30 }
31
32 contract StandardToken is Token {
33     function transfer(address _to, uint256 _value) returns (bool success) {
34         //Default assumes totalSupply can't be over max (2^256 - 1).
35         //If your token leaves out totalSupply and can issue more tokens as time goes on, you need to check if it doesn't wrap.
36         //Replace the if with this one instead.
37         //if (balances[msg.sender] >= _value && balances[_to] + _value > balances[_to]) {
38             if (balances[msg.sender] >= _value && _value > 0) {
39                 balances[msg.sender] -= _value;
```

Step 4. Deploy the contract

- 1. Switch to Deploy & Run tab
- 2. Choose Injected Web3 Environment
- 3. Deploy your token using your account address!
 - You should be logged in with your MetaMask; a new screen will ask you to confirm the contract



Step 4. Deploy the contract

- 4. The deployed contract should show up on your MetaWallet. View it on Etherscan and obtain the contract ID.

Timestamp: 3 mins ago (Jan-21-2020 04:34:27 PM +UTC)

From: [0xf7a9d2f8190ccf0421d87fd613dc795d0b69f4a2](#)

To: [Contract [0xbbef8aa56e0aa14cf9b827d9997f5c7769d064f1](#) Created] ✓

Value: 0 Ether (\$0.00)



Ropsten Test Network

History

#48 - 1/21/2020 at 11:34

Contract Deploym... -0 ETH
CONFIRMED

Details

From: 0xf7a9d2f8190CcF0... > New Contract

Transaction	
Amount	0 ETH
Gas Limit (Units)	1045754
Gas Used (Units)	1045754
Gas Price (GWEI)	1
Total	0.001046 ETH

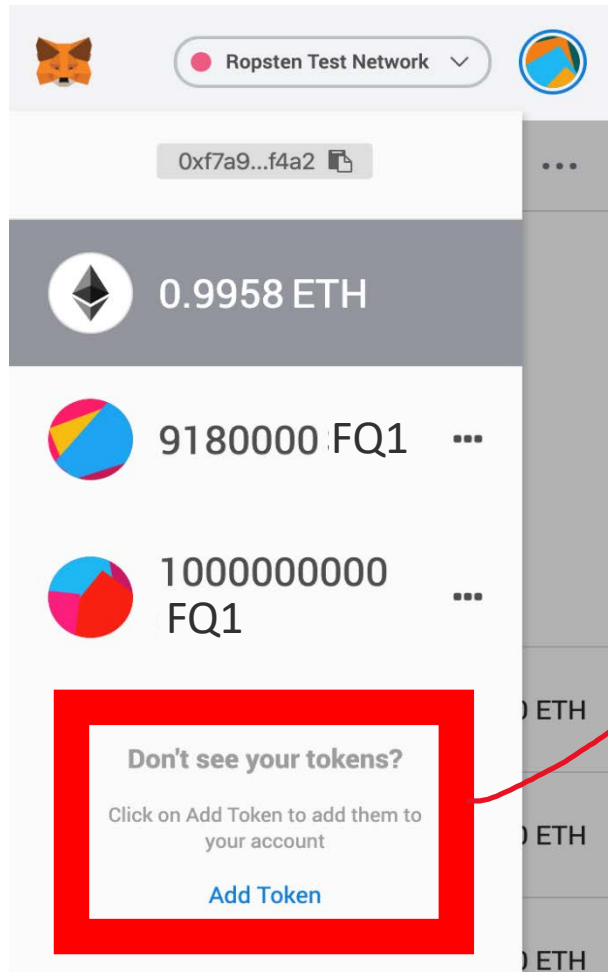
Activity Log

- Transaction created with a value of 0 ETH at 11:34 on 1/21/2020.
- Transaction submitted with gas fee of 0 WEI at 11:34 on 1/21/2020.

Look carefully:

- The contract ID is not your public address

Step 5. Add tokens



Add Tokens

Search [Custom Token](#)

Token Contract Address
0xbbef8aa56e0aa14cf9b827d9997f5c7769d06

Token Symbol [Edit](#)
SYM

Decimals of Precision
0

[Cancel](#) [Next](#)

The contract ID
(It is **not** your public address)

Symbol should match the one you establish in the code

Step 6. Using the tokens

You can send tokens to anyone with a MetaMask address

- They will have to follow Step 5 to add the custom token to their wallet.

Further information

The Solidity code is from:

- <https://github.com/CodeWithJoe2020/ERC20Token/blob/main/ERC20.sol>