Blockchain

Smart Contract
&
Solidity

2024

目錄

- Introduction to Smart Contract
- EVM 與智能合約
- Solidity

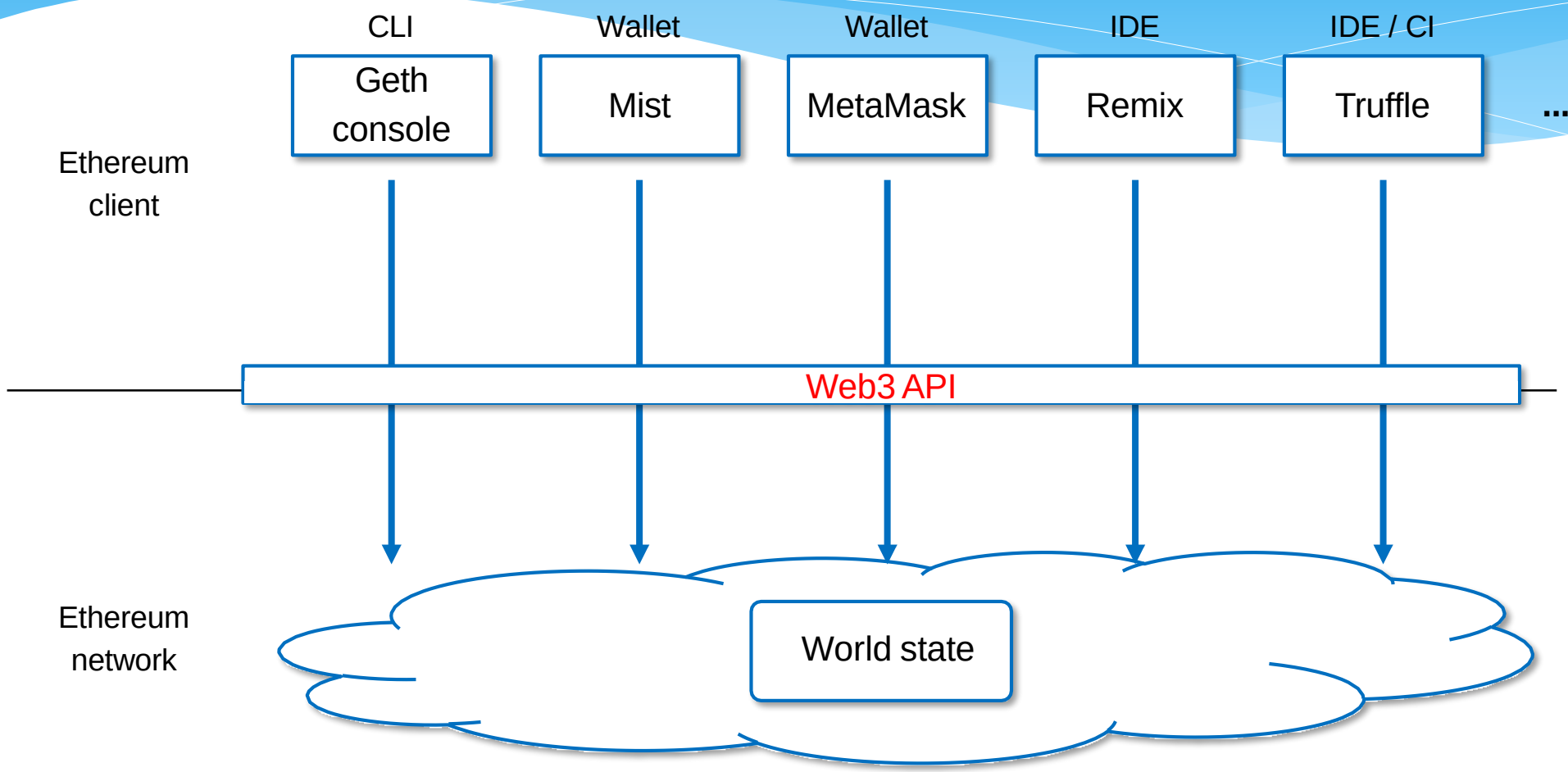
Introduction to Smart Contract

A Simple Smart Contract

- Storage Example。
- Line 1 : source code 使用GPL 3.0
- Line 2 : 使用solidity 0.8.26以上或0.9.0以下版本
- Contract : Solidity 內的contract是保存在以太坊區塊鏈上特定位址的程式碼（功能）和資料（狀態）的集合。
- 合約定義了可用於修改或擷取變數值的函數 set 和 get 。

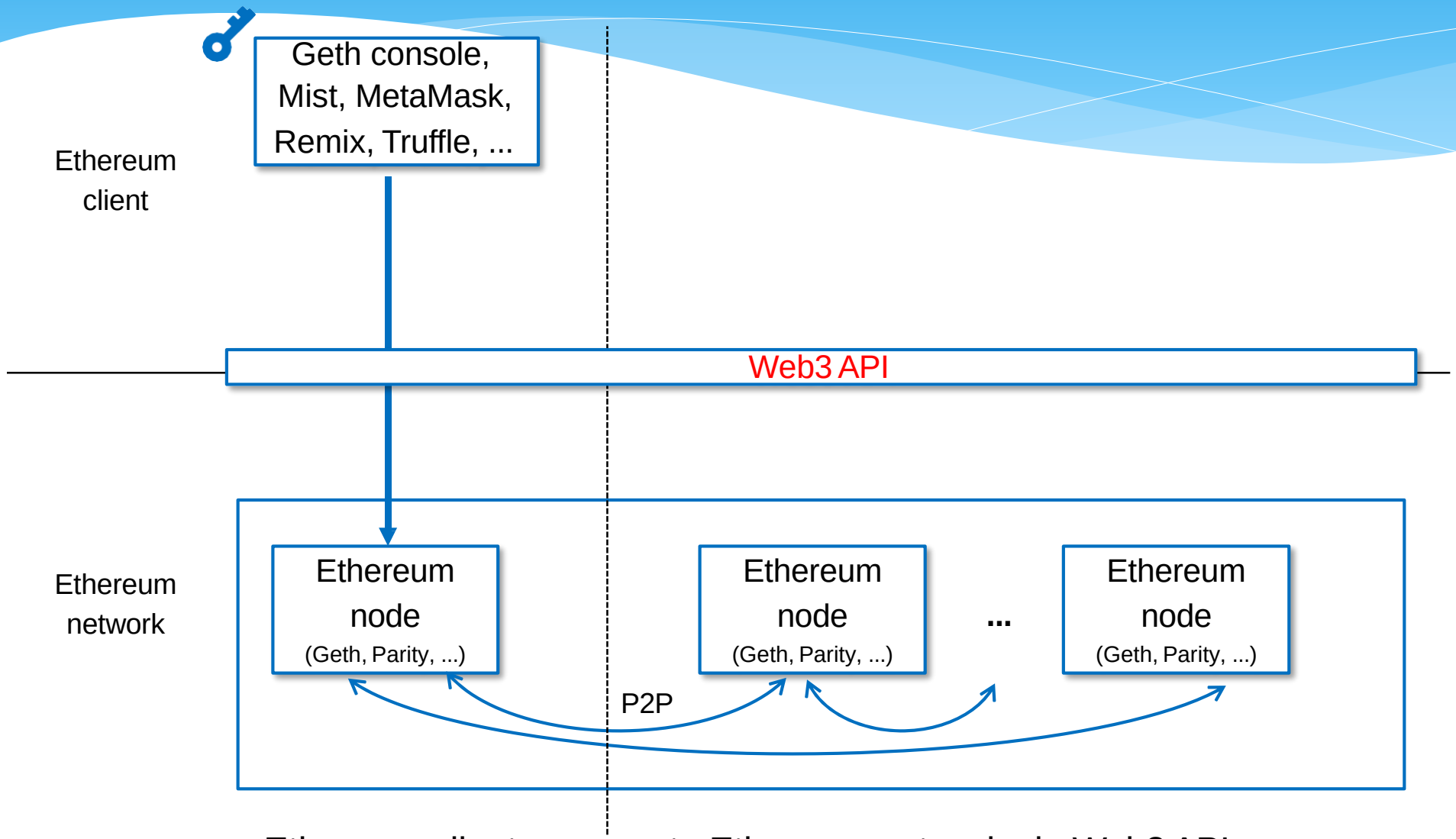
```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.26 <0.9.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

    function get() public view returns (uint) {
        return storedData;
    }
}
```

Web3 API and client



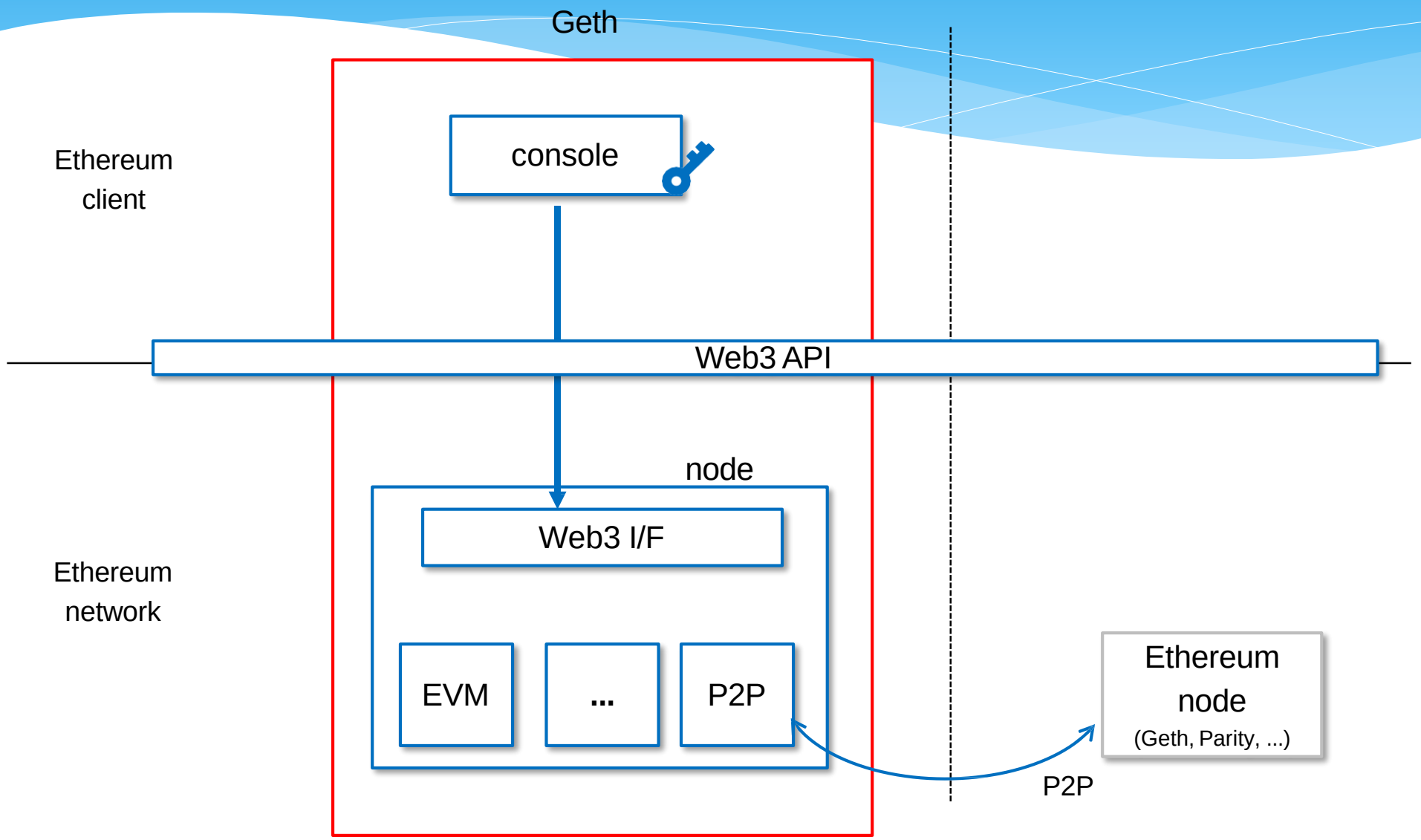
Ethereum clients access to Ethereum network via Web3 API.

Web3 API and client

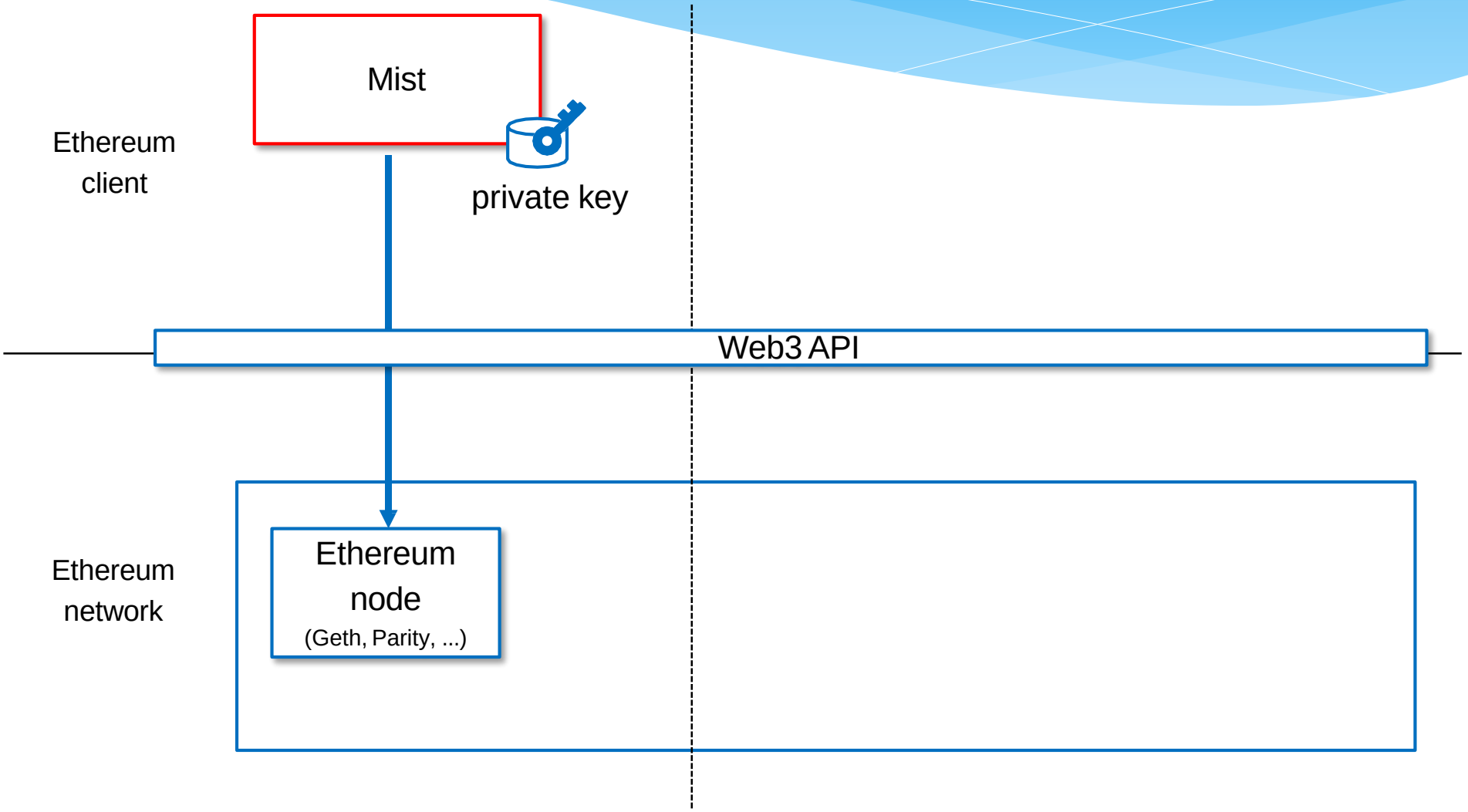


Ethereum clients access to Ethereum network via Web3 API.

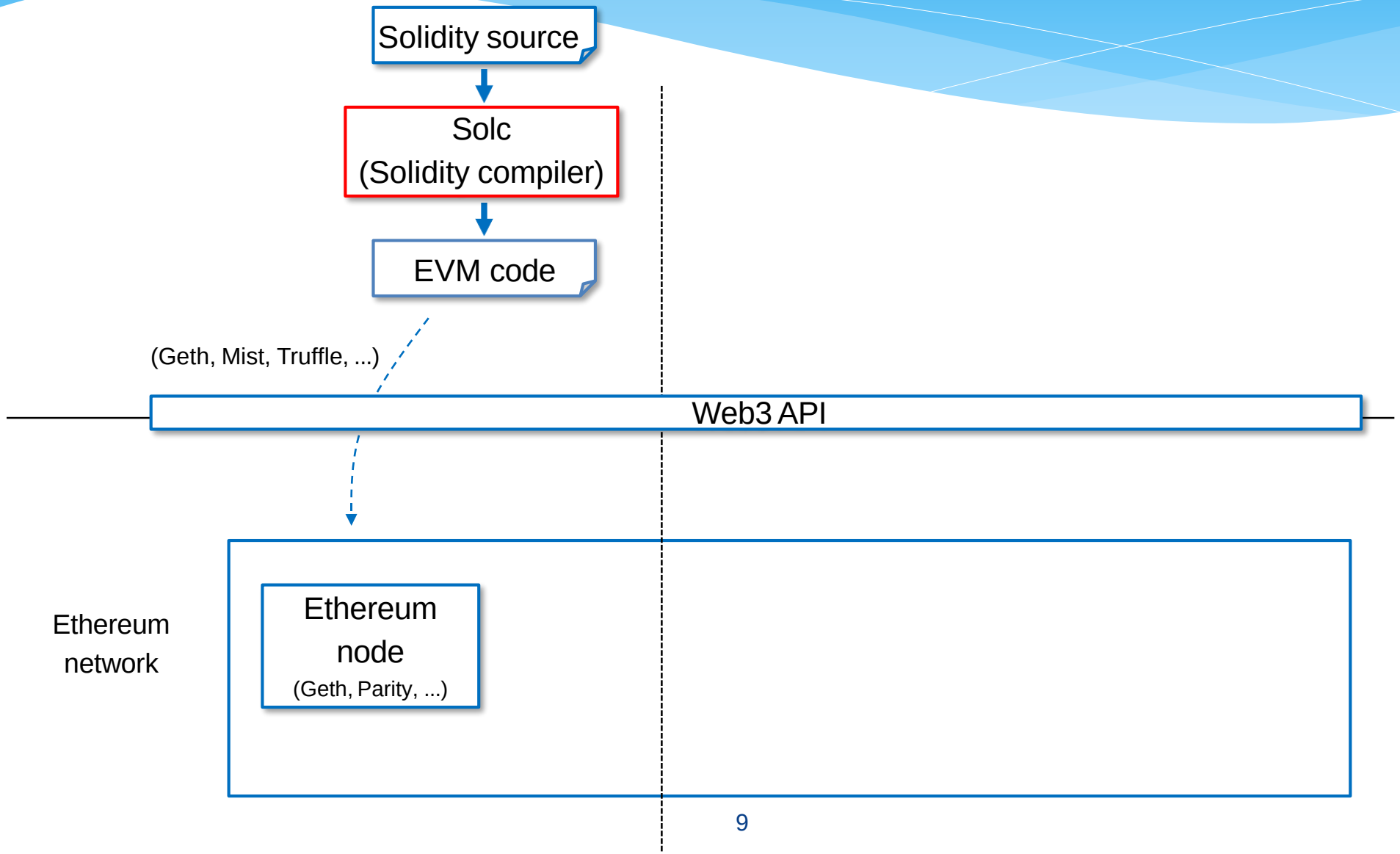
Geth


Geth
Ethereum client
console
Web3 API
node
Web3 I/F
Ethereum network
EVM
...
P2P
Ethereum node
(Geth, Parity, ...)
P2P

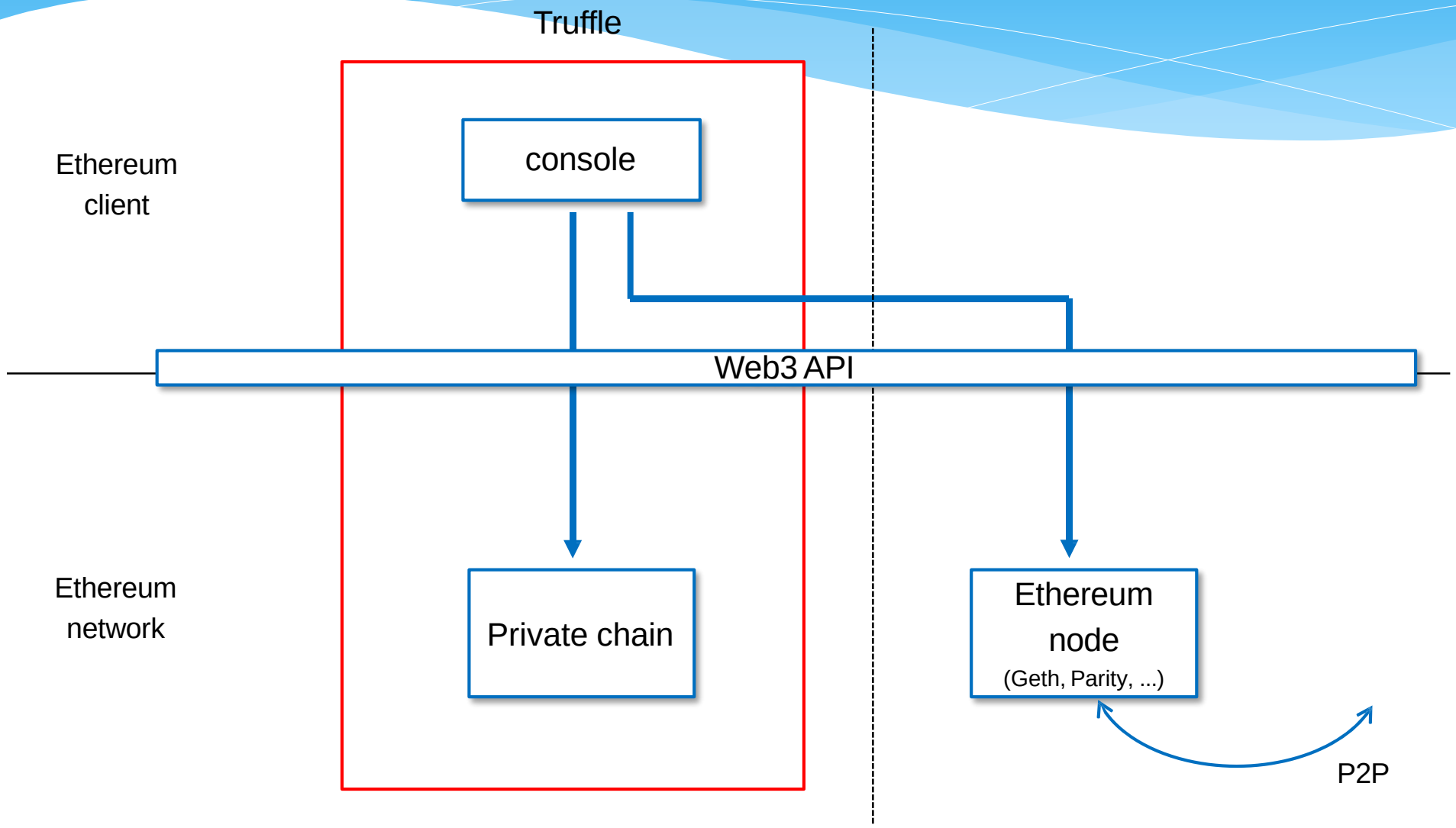
Mist


Mist
Ethereum client
private key
Web3 API
Ethereum network
Ethereum node
(Geth, Parity, ...)

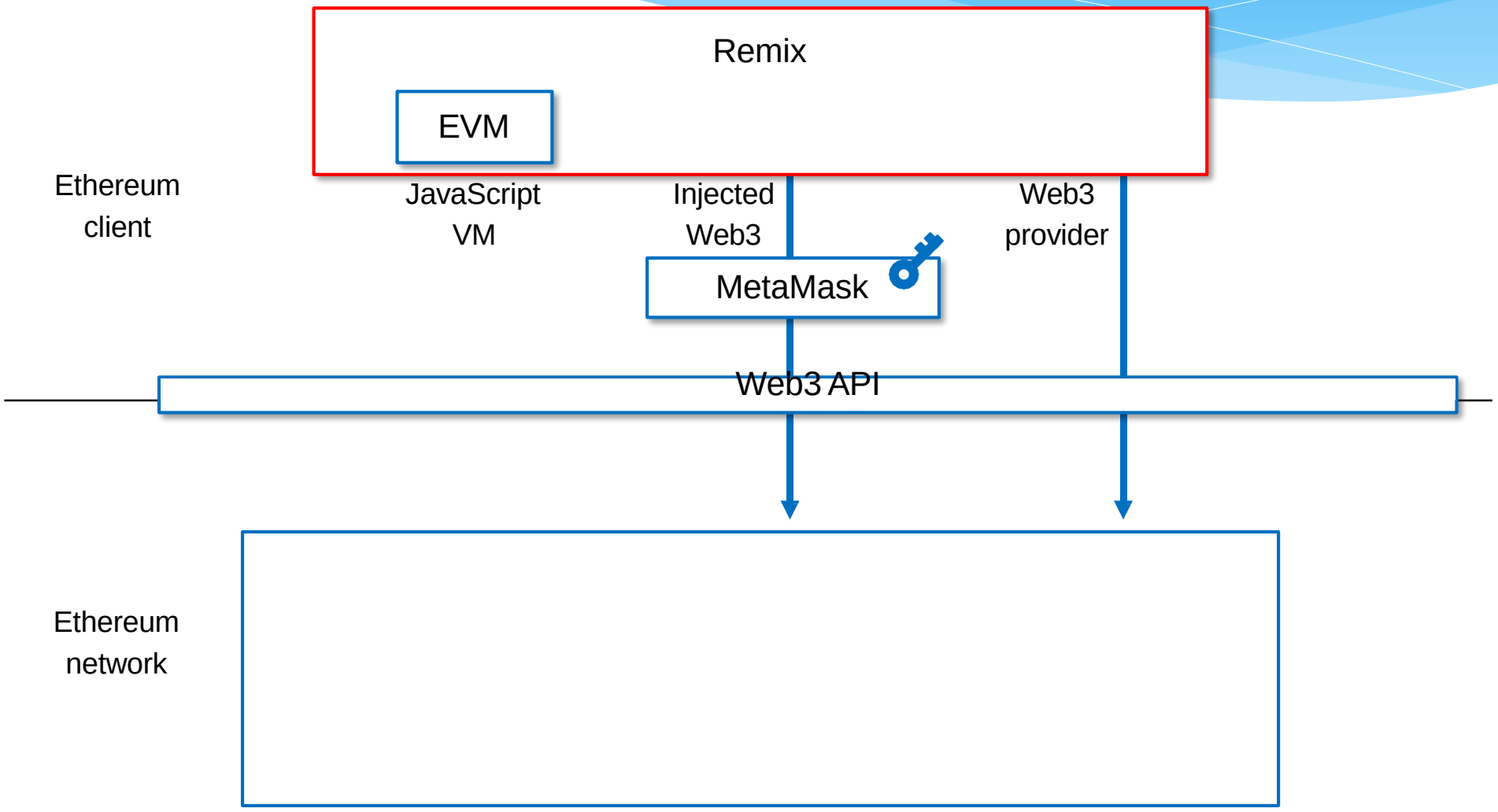
Solc


Solidity source
Solc
(Solidity compiler)
EVM code
(Geth, Mist, Truffle, ...)
Web3 API
Ethereum
network
Ethereum
node
(Geth, Parity, ...)

Truffle


Truffle
Ethereum client
console
Web3 API
Ethereum network
Private chain
Ethereum node
(Geth, Parity, ...)
P2P

Remix


Remix
EVM
Ethereum client
JavaScript VM
Injected Web3
Web3 provider
MetaMask
Web3 API
Ethereum network

Start to Deploy Smart Contract

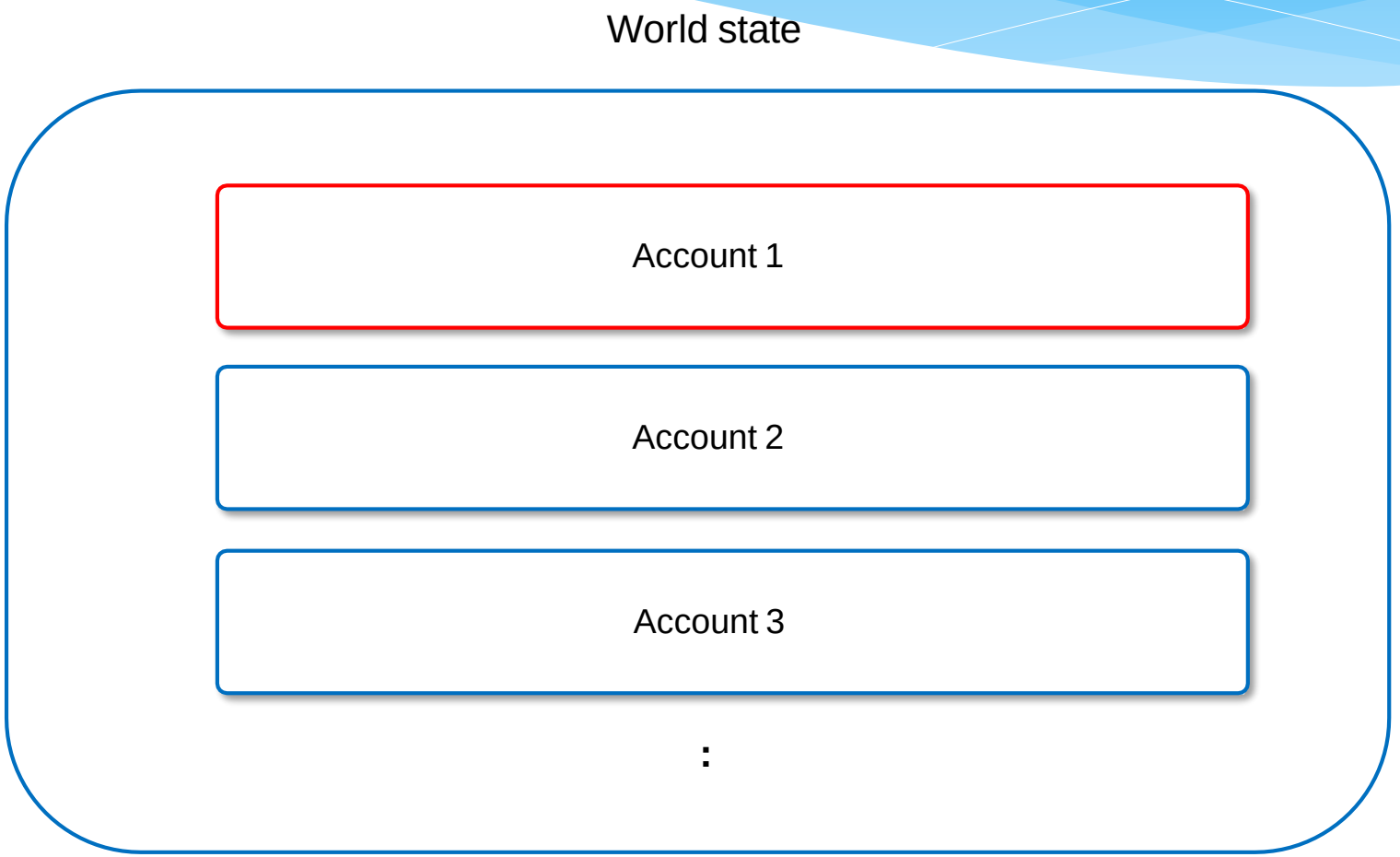
61

EVM 與智能合約

EVM

- 以太坊虛擬機（EVM）是以太坊中智慧合約的運行時環境。
- 它是沙箱，而且實際上是完全隔離的，
- EVM 內運行的程式碼無法存取網路、檔案系統或其他程序。
- 智能合約甚至對其他智能合約的存取也受到限制。

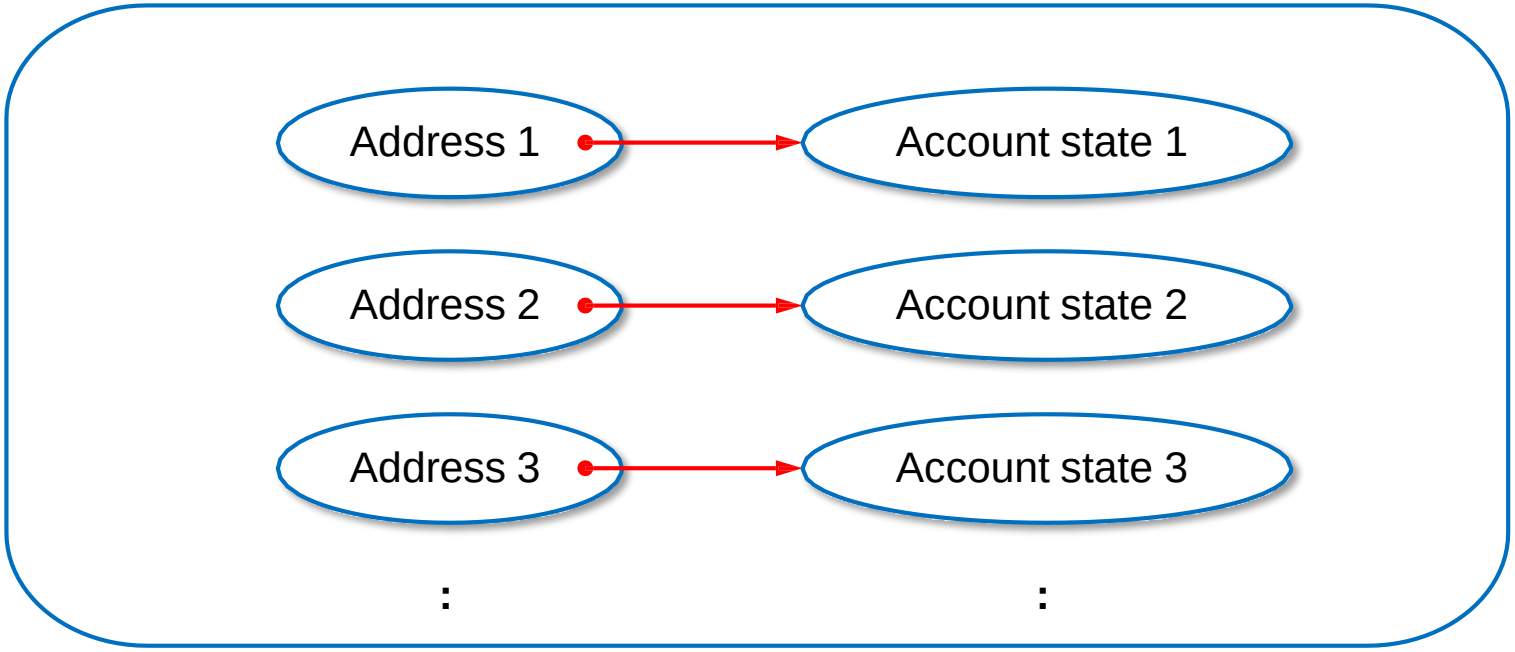
Account



An account is an object in the world state.

World state

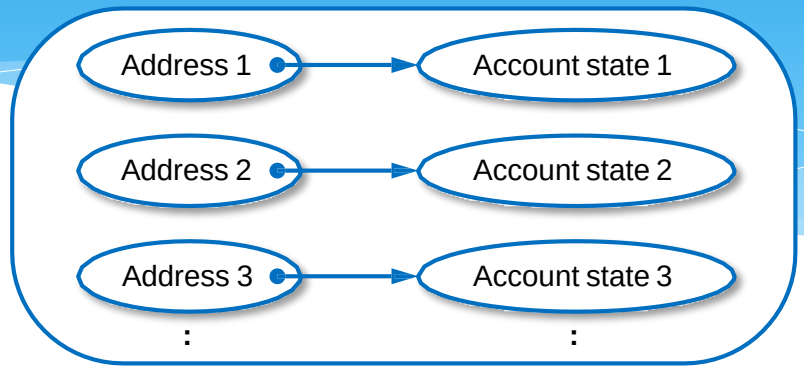
World state σ_t



1. The world state can be seen as the global state that is constantly updated by transaction executions.
2. The world state is a mapping between address and account state.

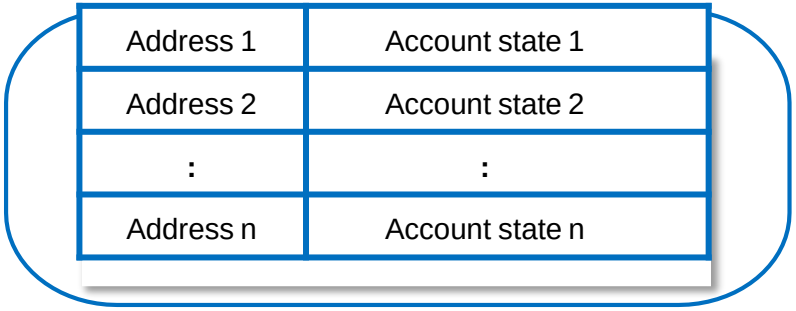
Several views of World state

Mapping view



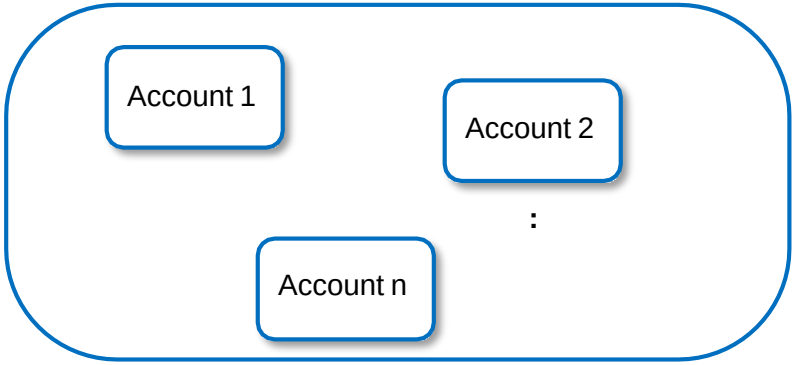
World state

Table view

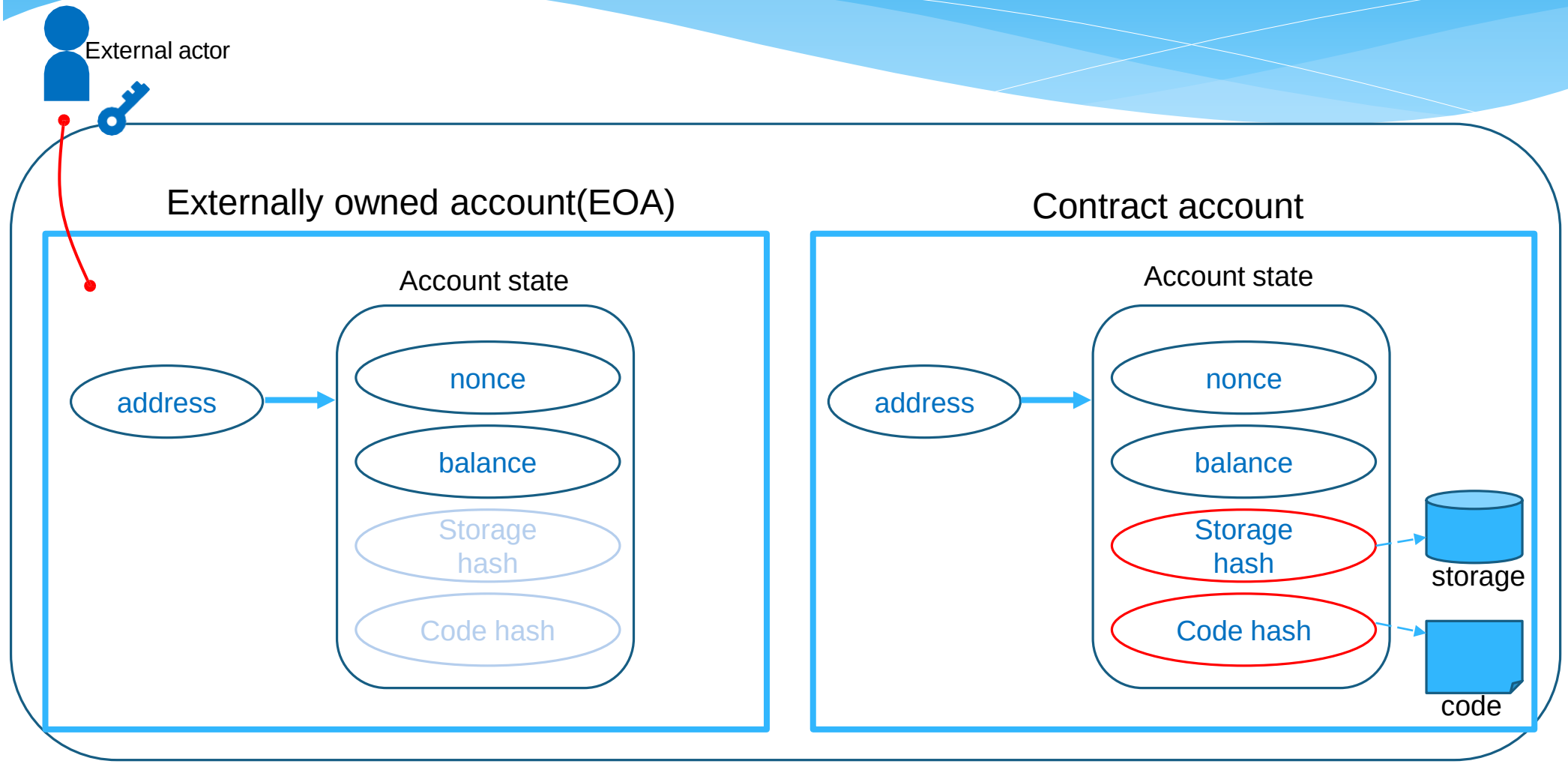


World state

Object view



Two types of account

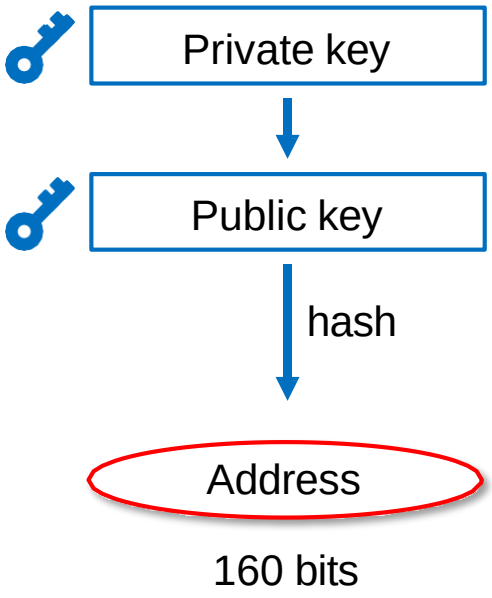


EOA is controlled by a private key.
EOA cannot contain EVM code.

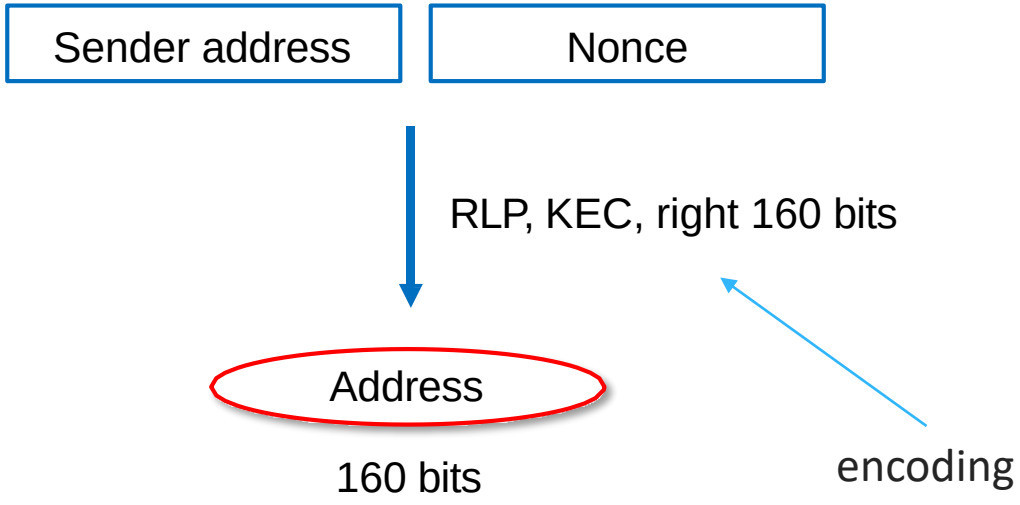
Contract contains EVM code.
Contract is controlled by EVM code.

Address of account

Externally owned account (EOA)

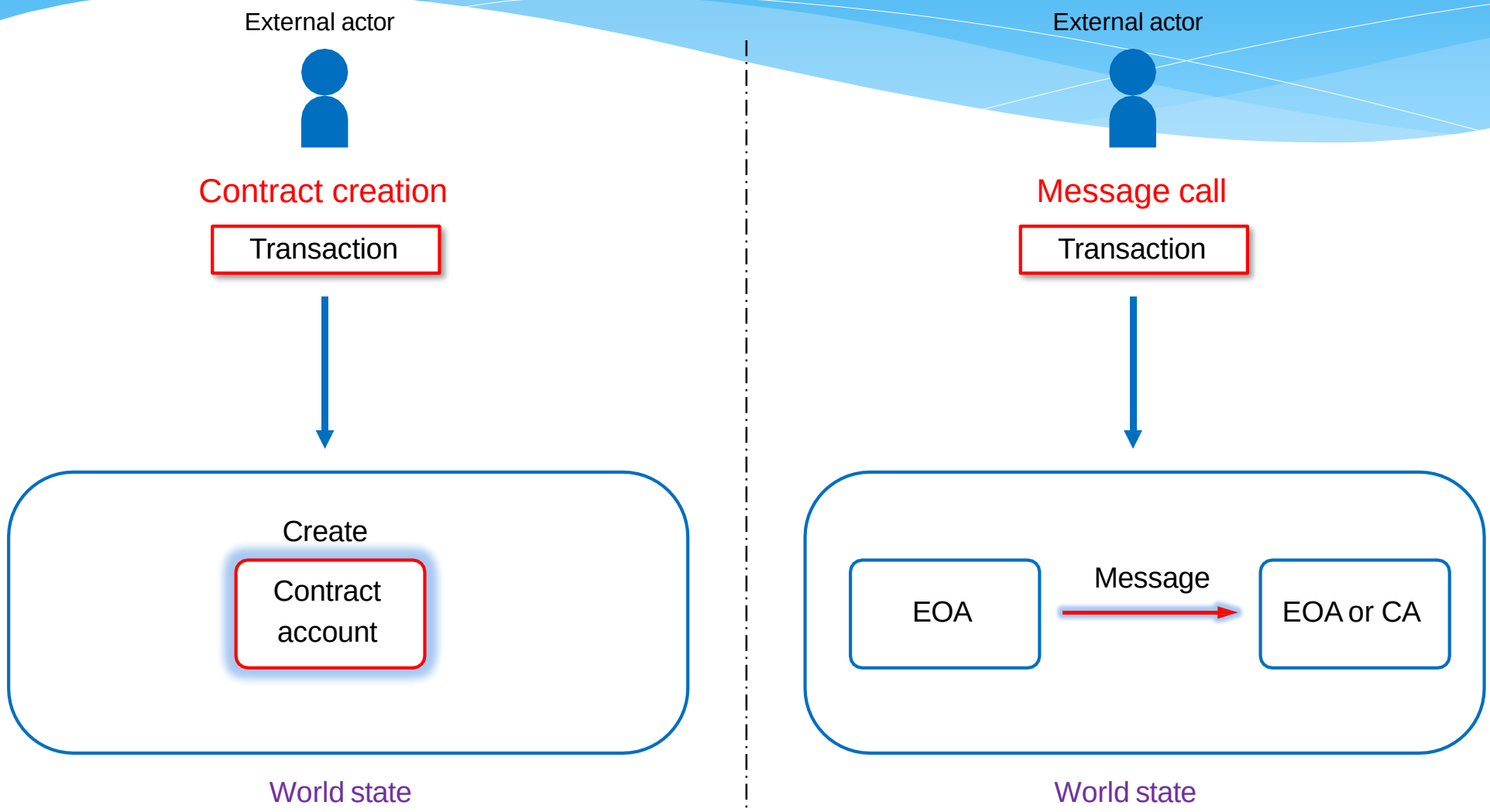


Contract account



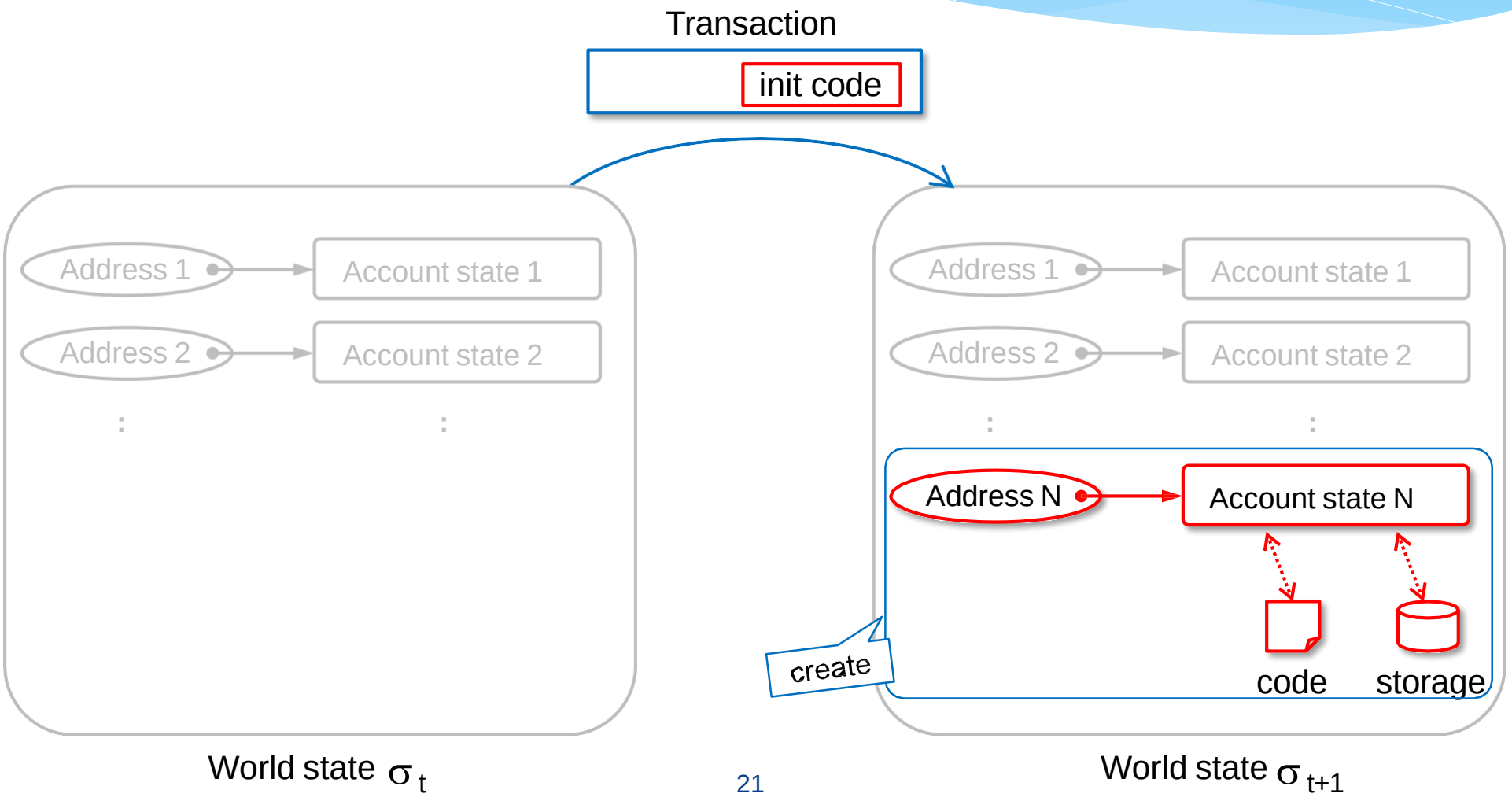
A 160-bit code used for identifying accounts.

Two types of transaction

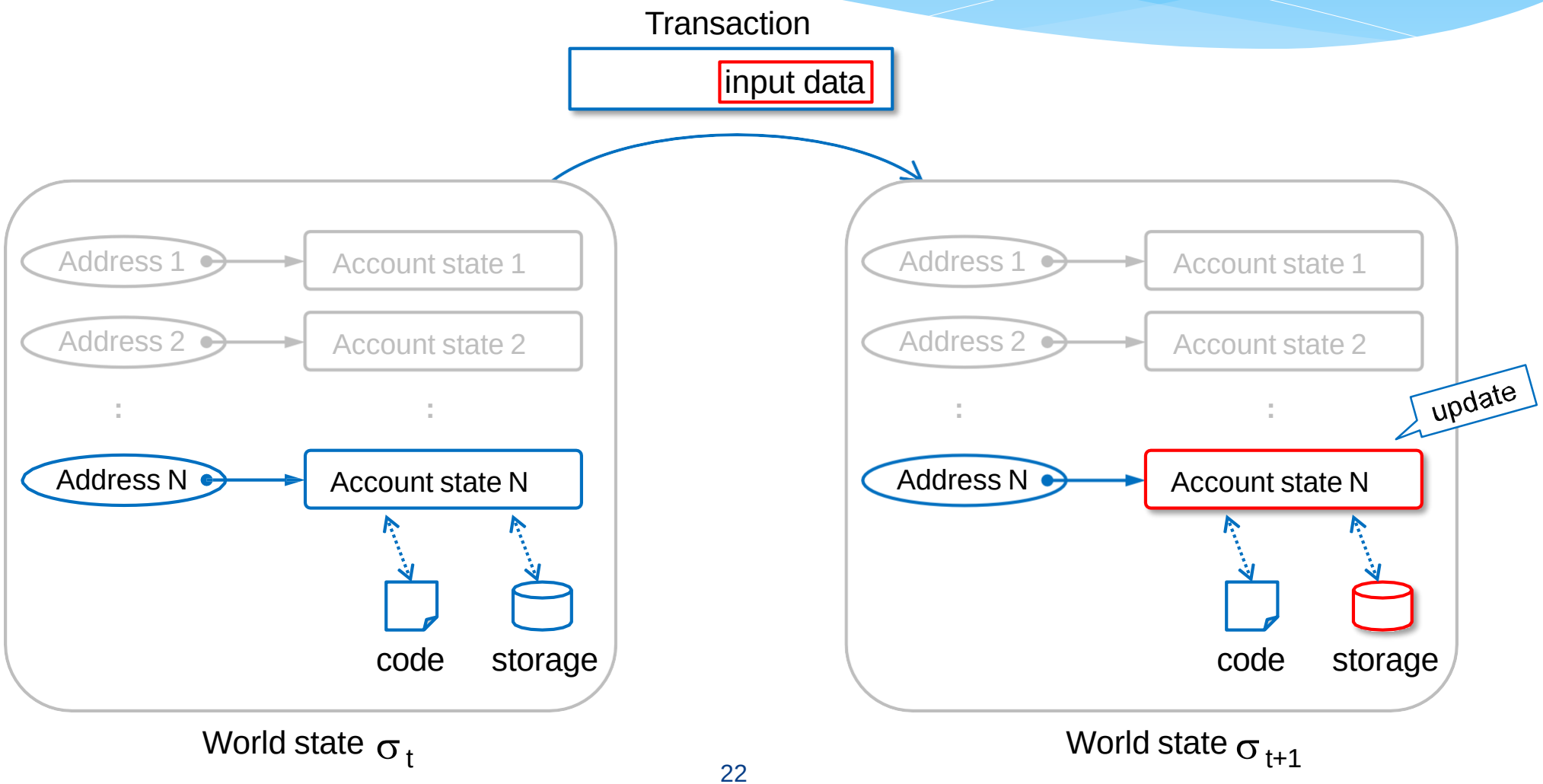


There are two practical types of transaction, contract creation and message call.

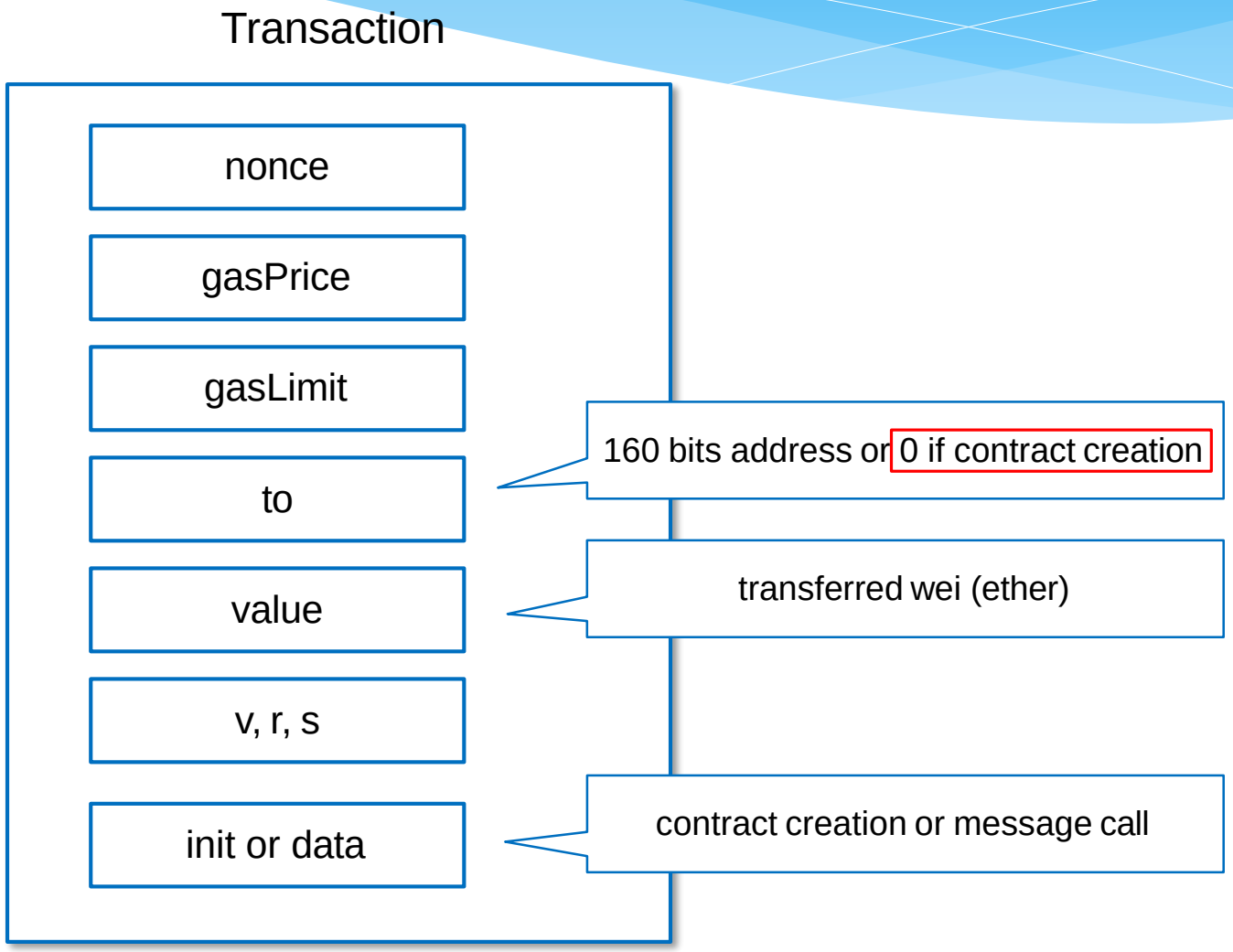
Contract creation



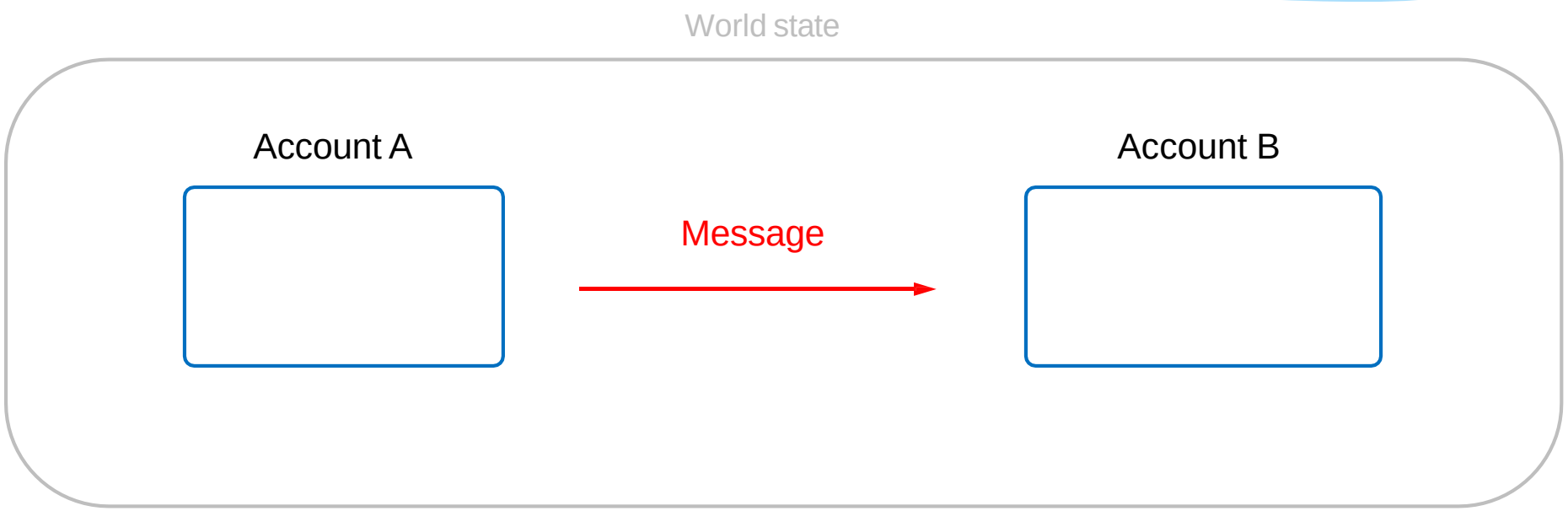
Message call



Field of a transaction



Message

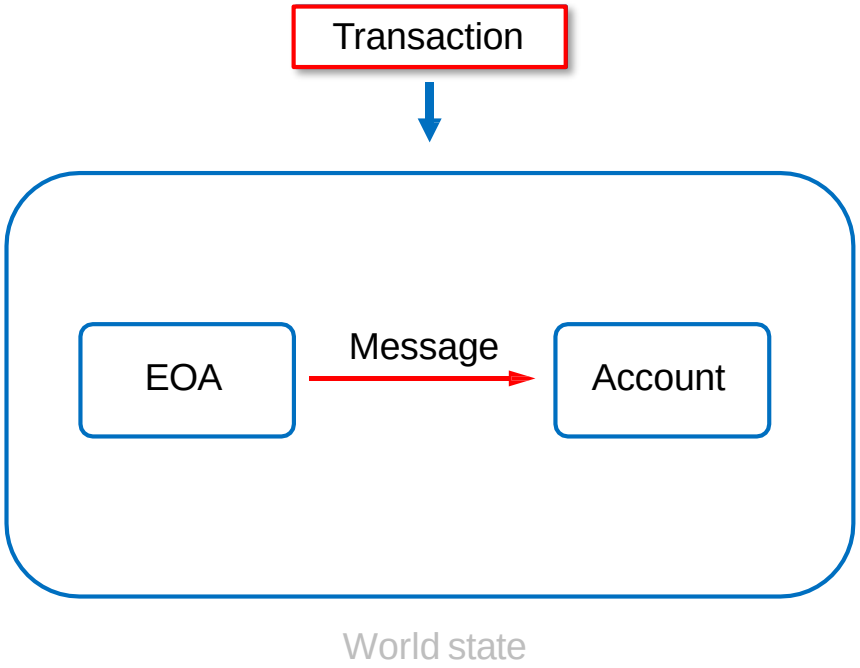


Message is passed between two Accounts.

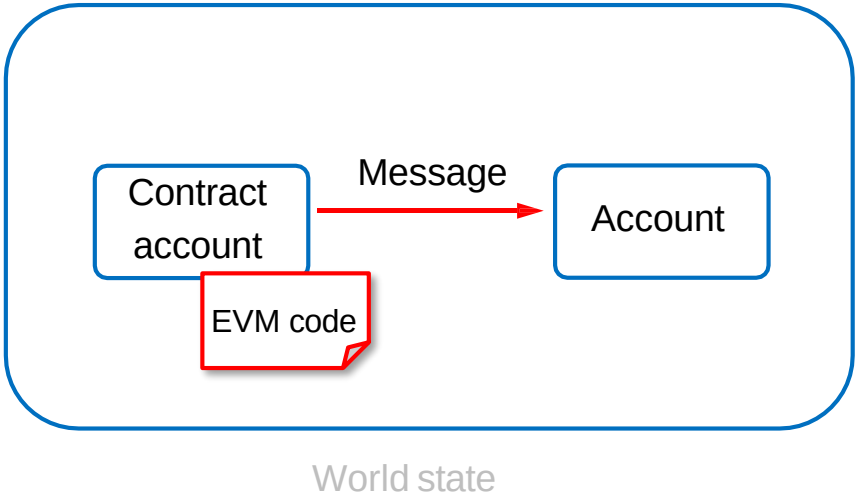
Message is Data (as a set of bytes) and Value (specified as Ether) .

Message

Triggered by transaction



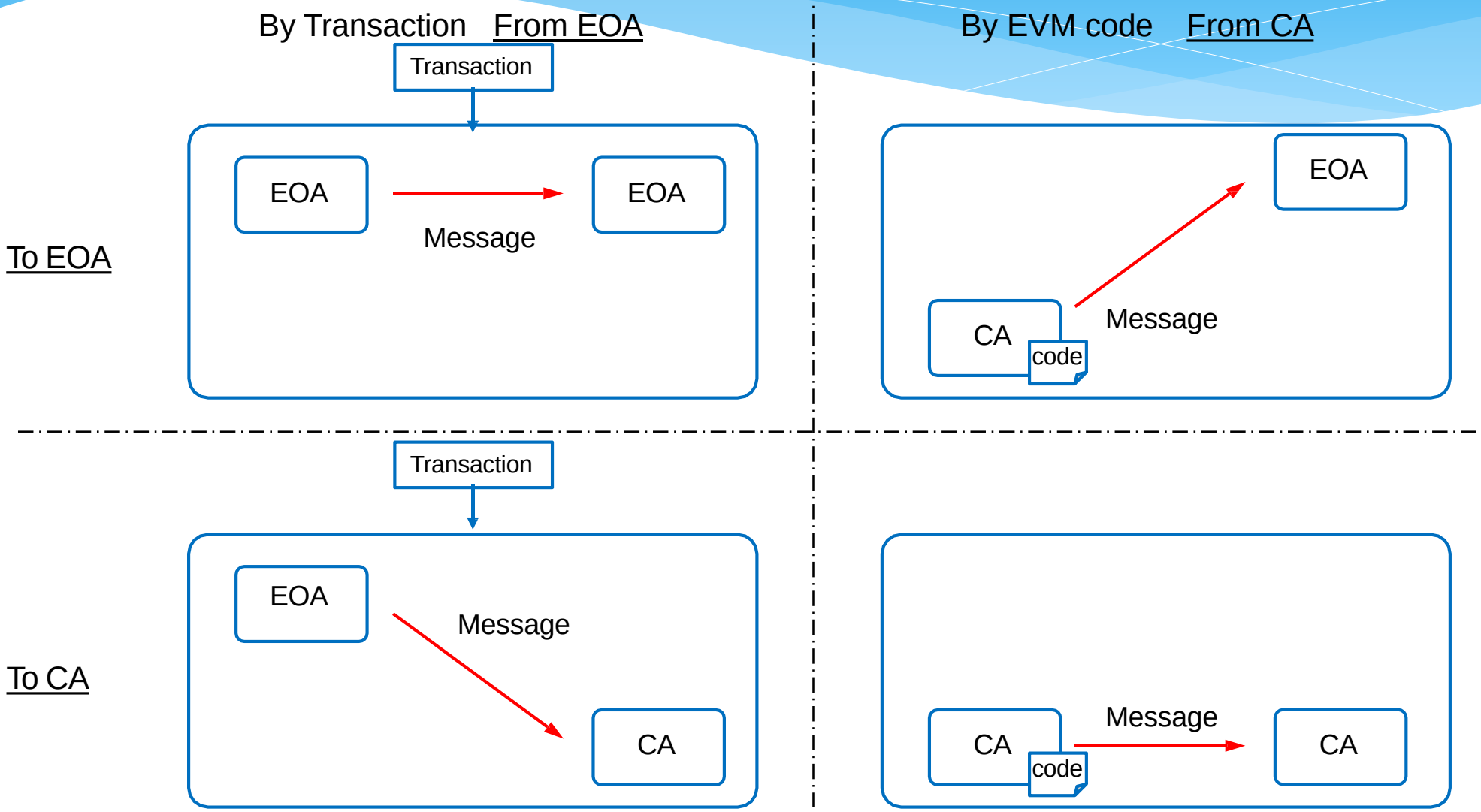
Triggered by EVM code



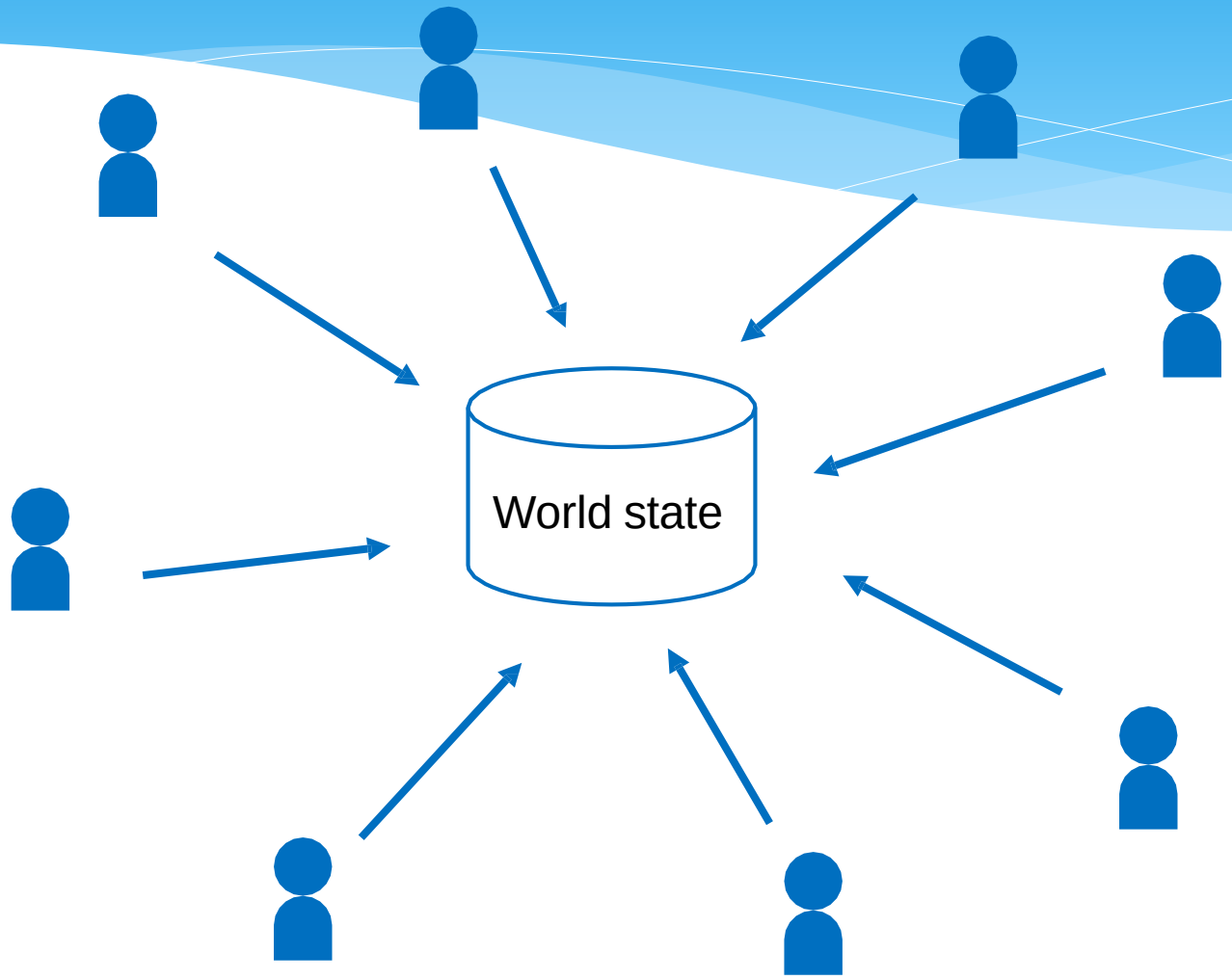
Transaction triggers an associated message.

EVM can also send a message.

Four cases of Message

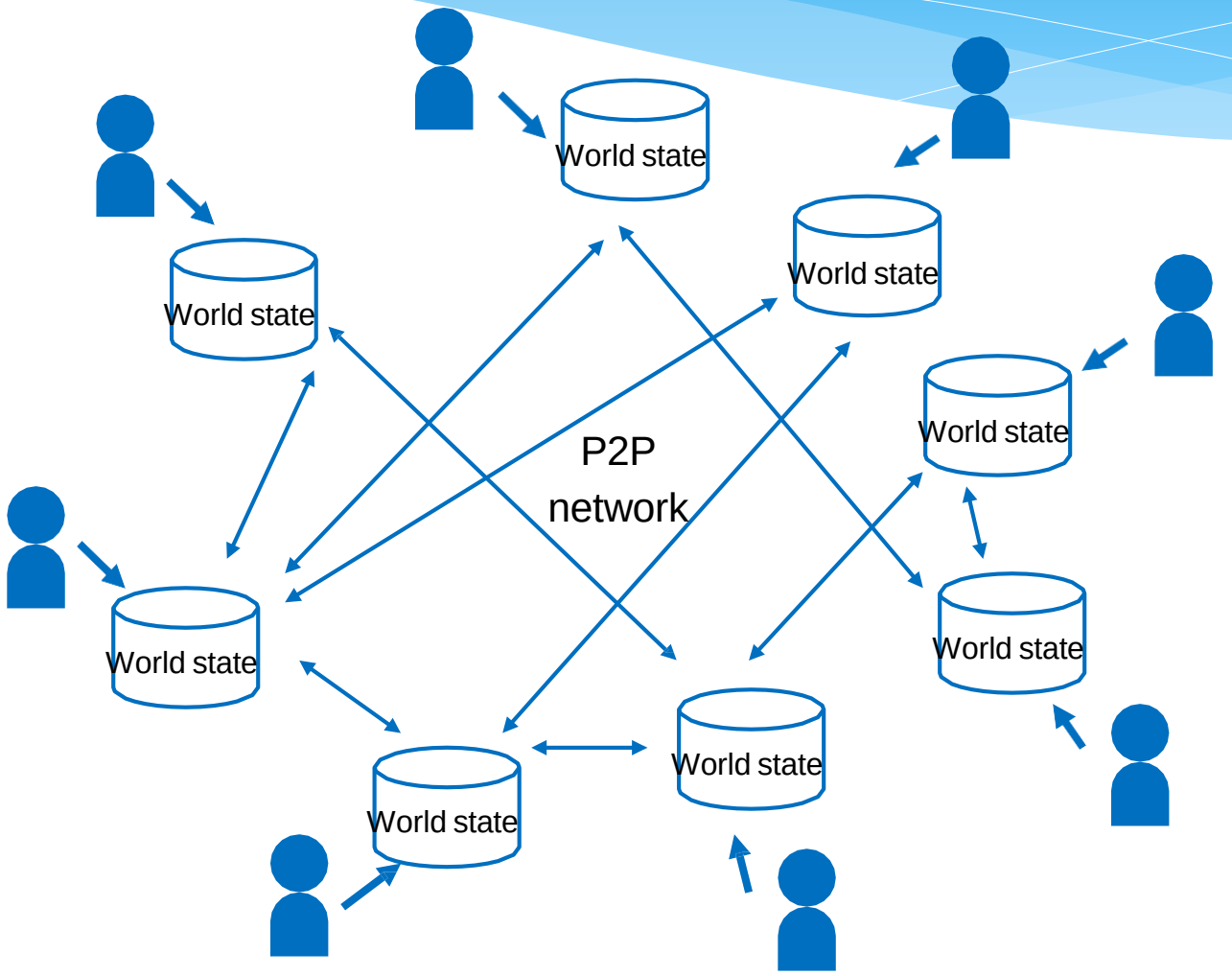


Globally shared, transactional database



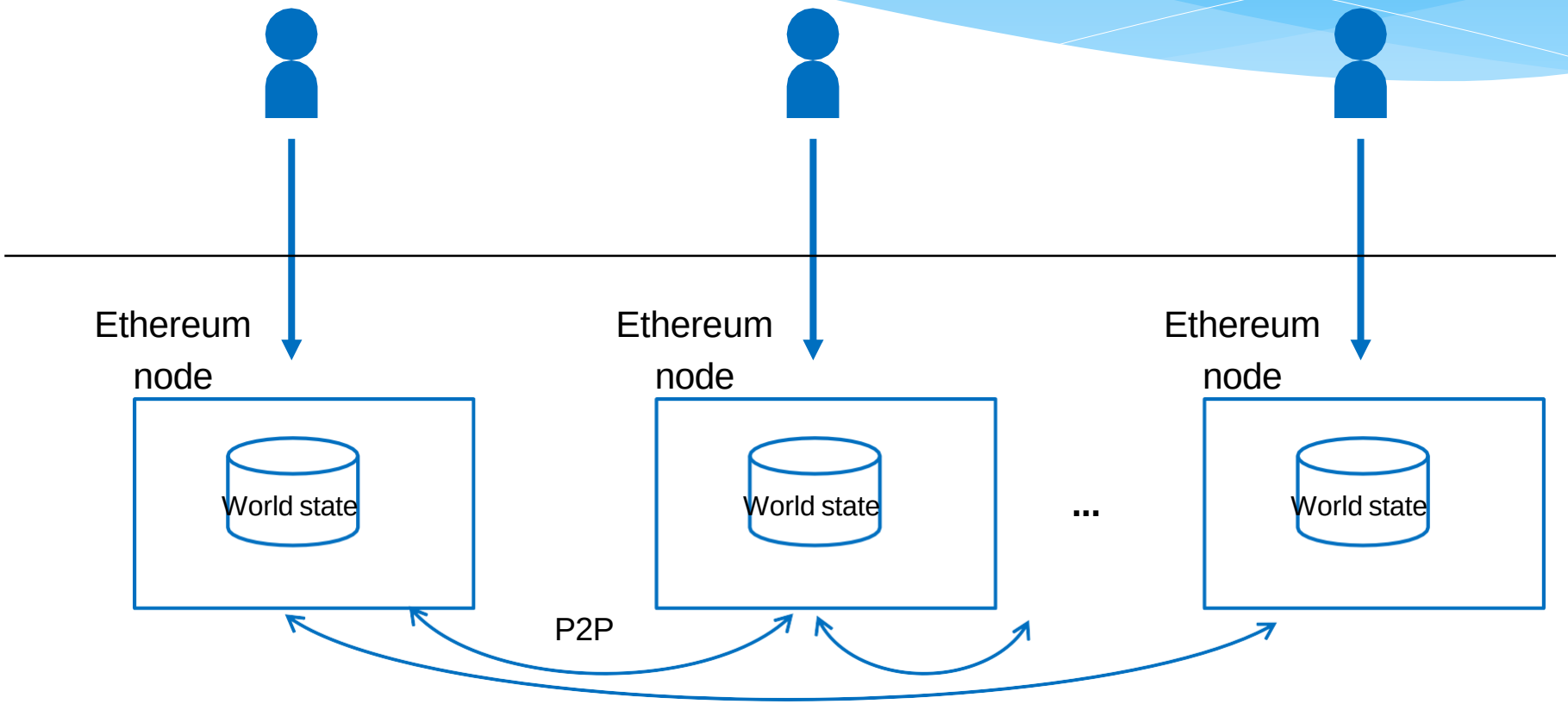
A blockchain is a globally shared, transactional database.

Decentralised database



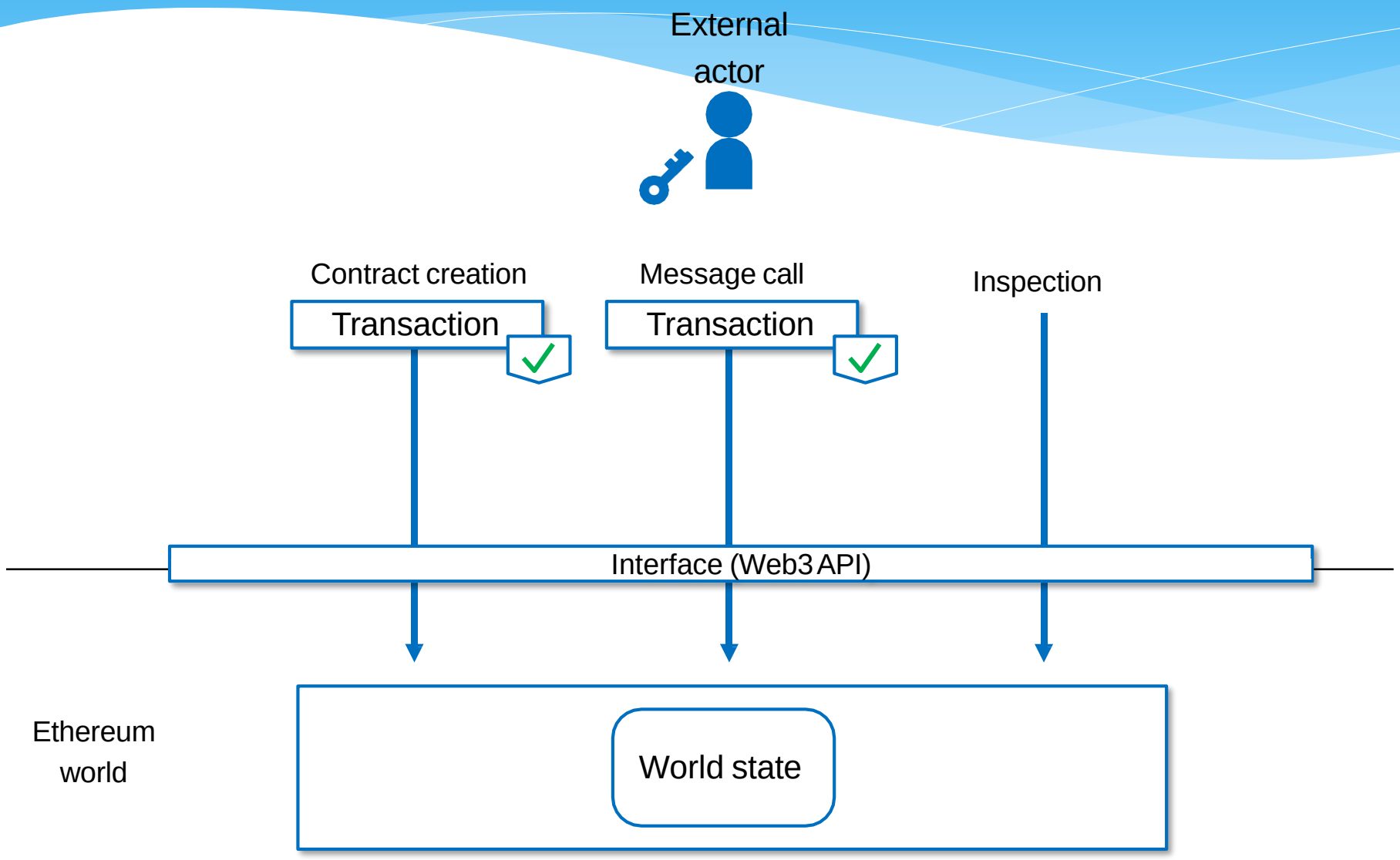
A blockchain is a globally shared, decentralised, transactional database.

P2P network inter nodes

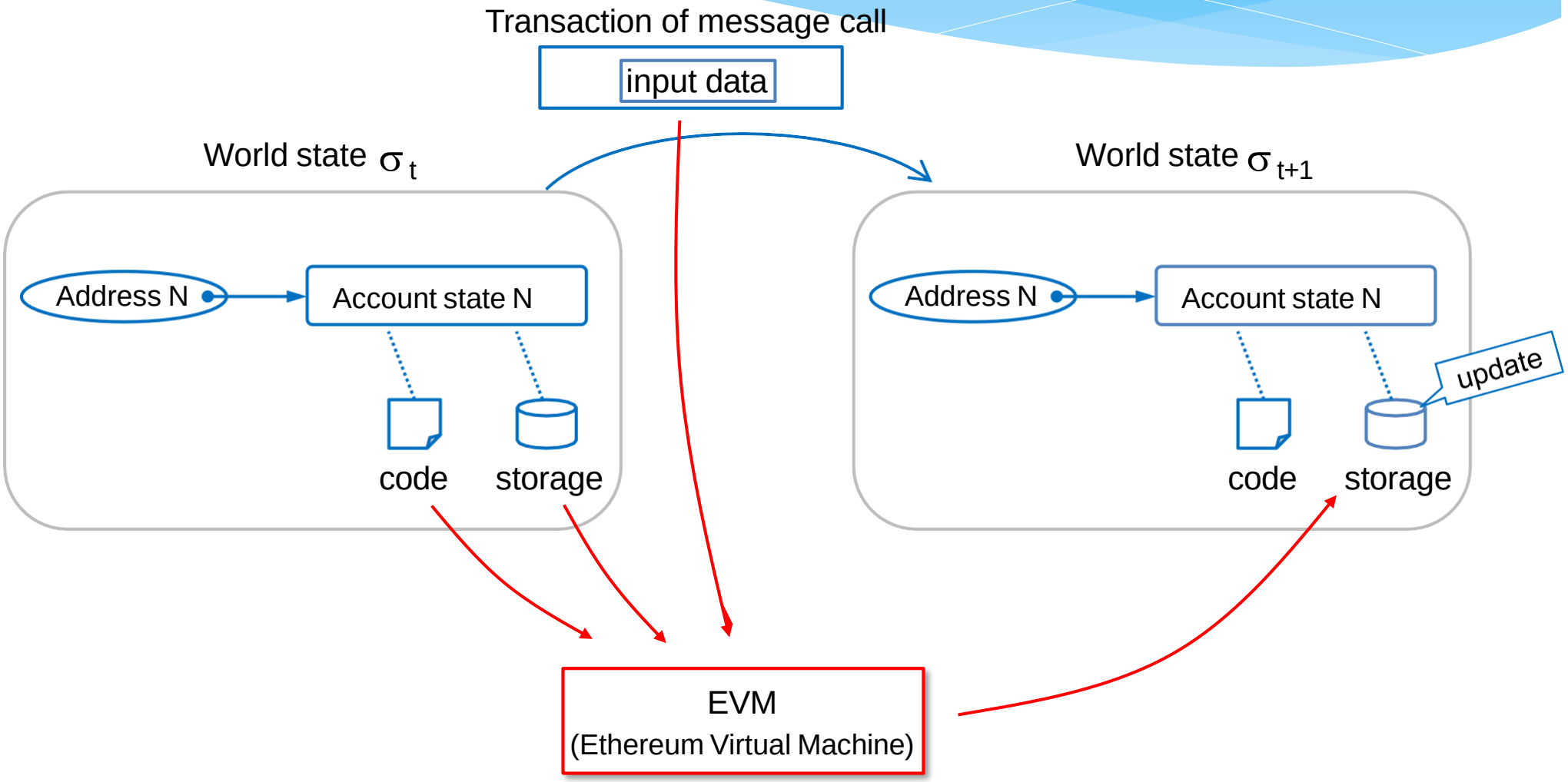


Decentralised nodes constitute Ethereum P2P network.

Interface to a node

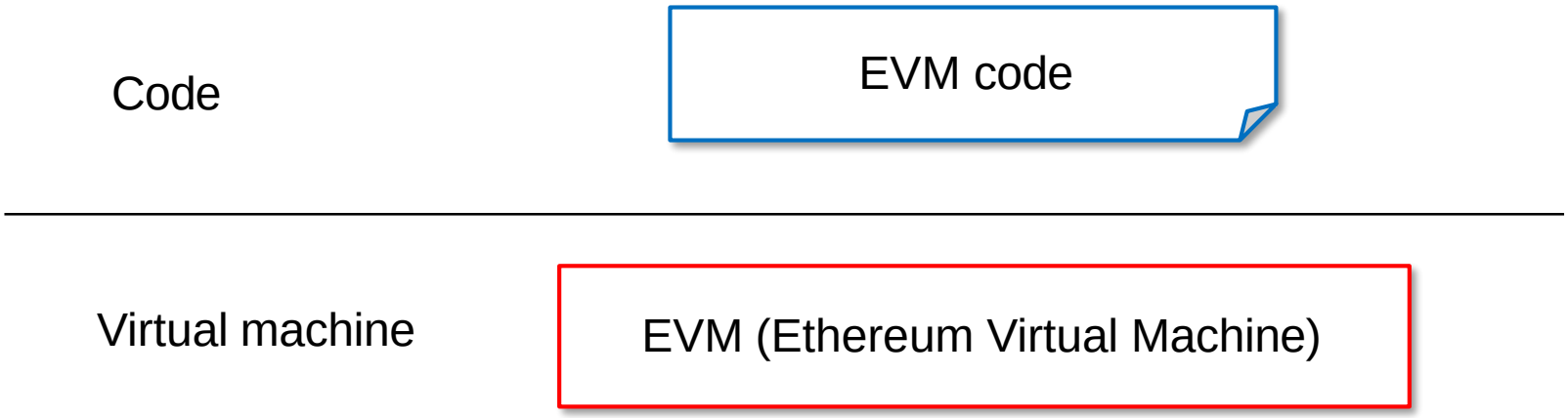


Ethereum virtual machine(EVM)



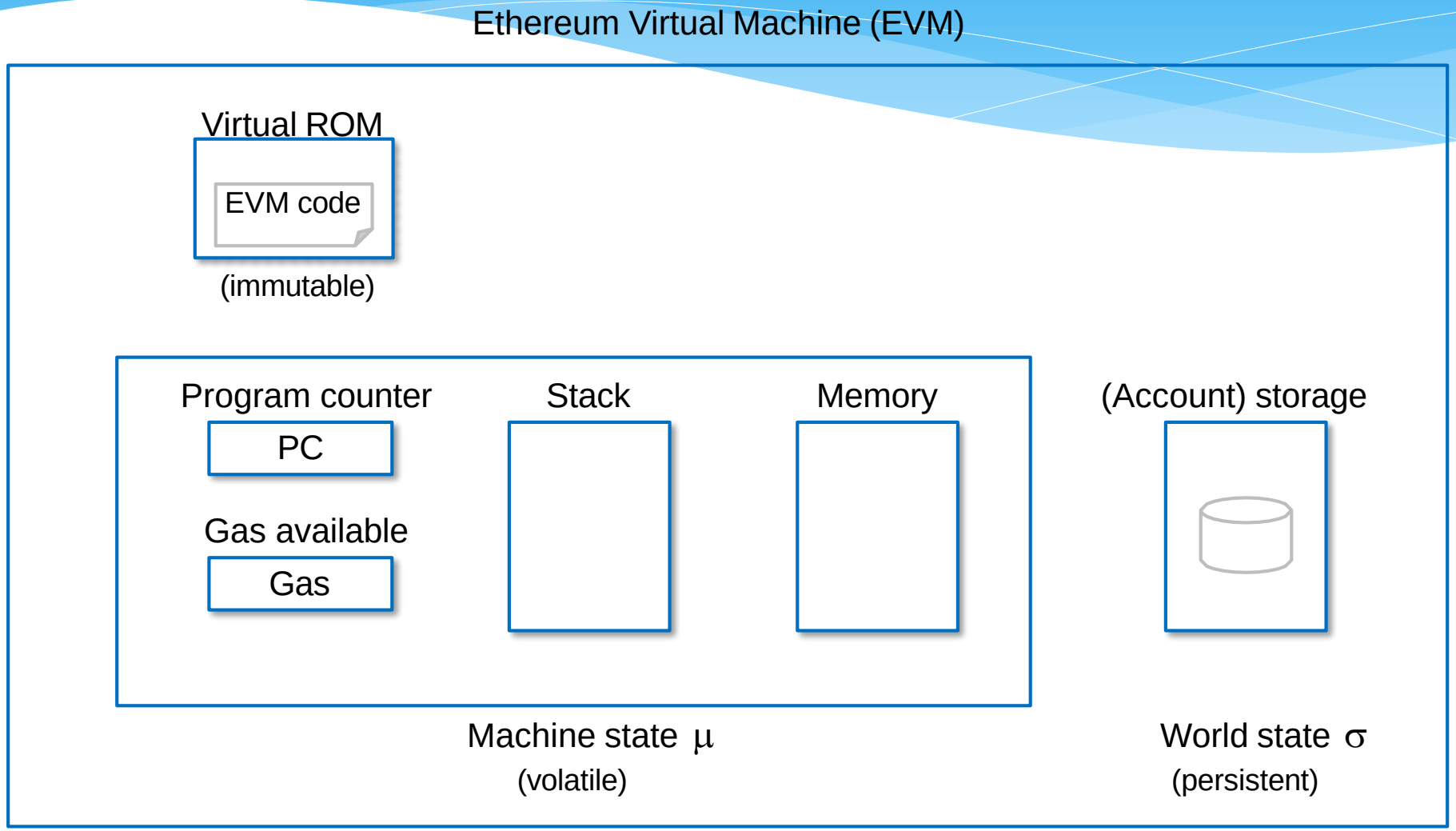
EVM code is executed on Ethereum Virtual Machine (EVM).

Ethereum virtual machine(EVM)



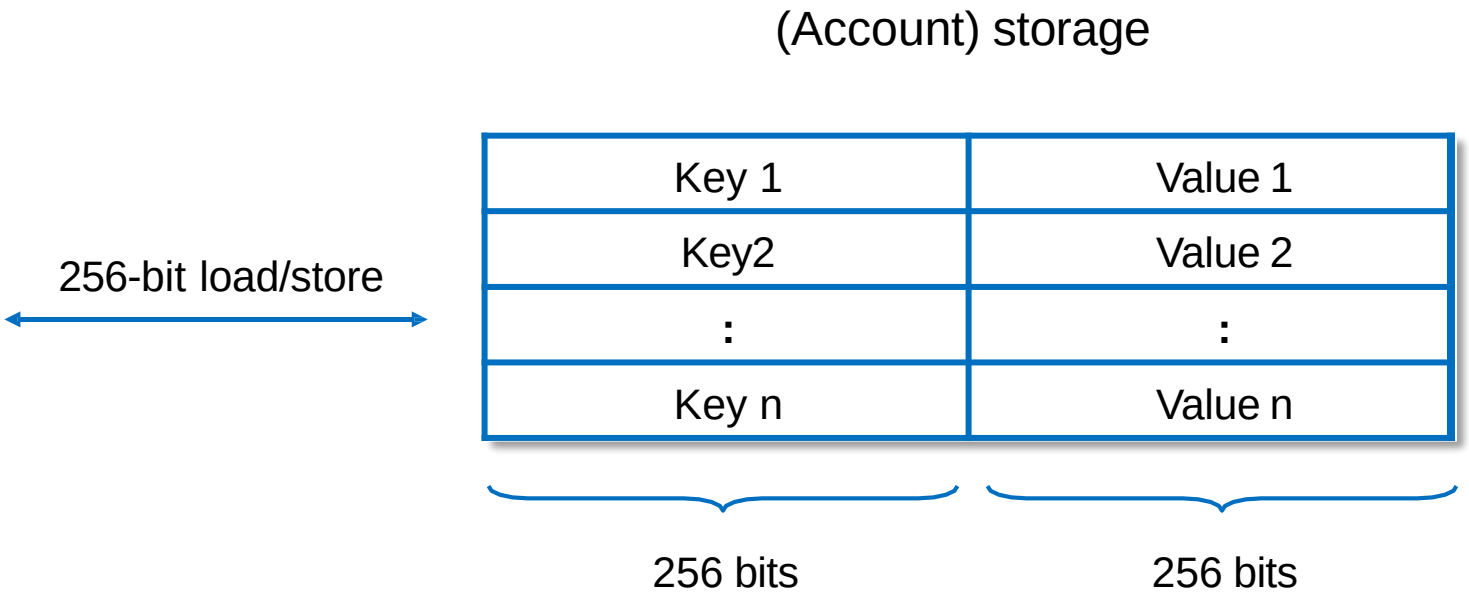
The Ethereum Virtual Machine is the runtime environment for smart contracts in Ethreum.

EVM architecture



The EVM is a simple stack-based architecture.

Account storage



Storage is a key-value store that maps 256-bit words to 256-bit words.

Access with SSTORE/SLOAD instructions.

All locations in storage are well-defined initially as zero.

EVM code

Assembly view

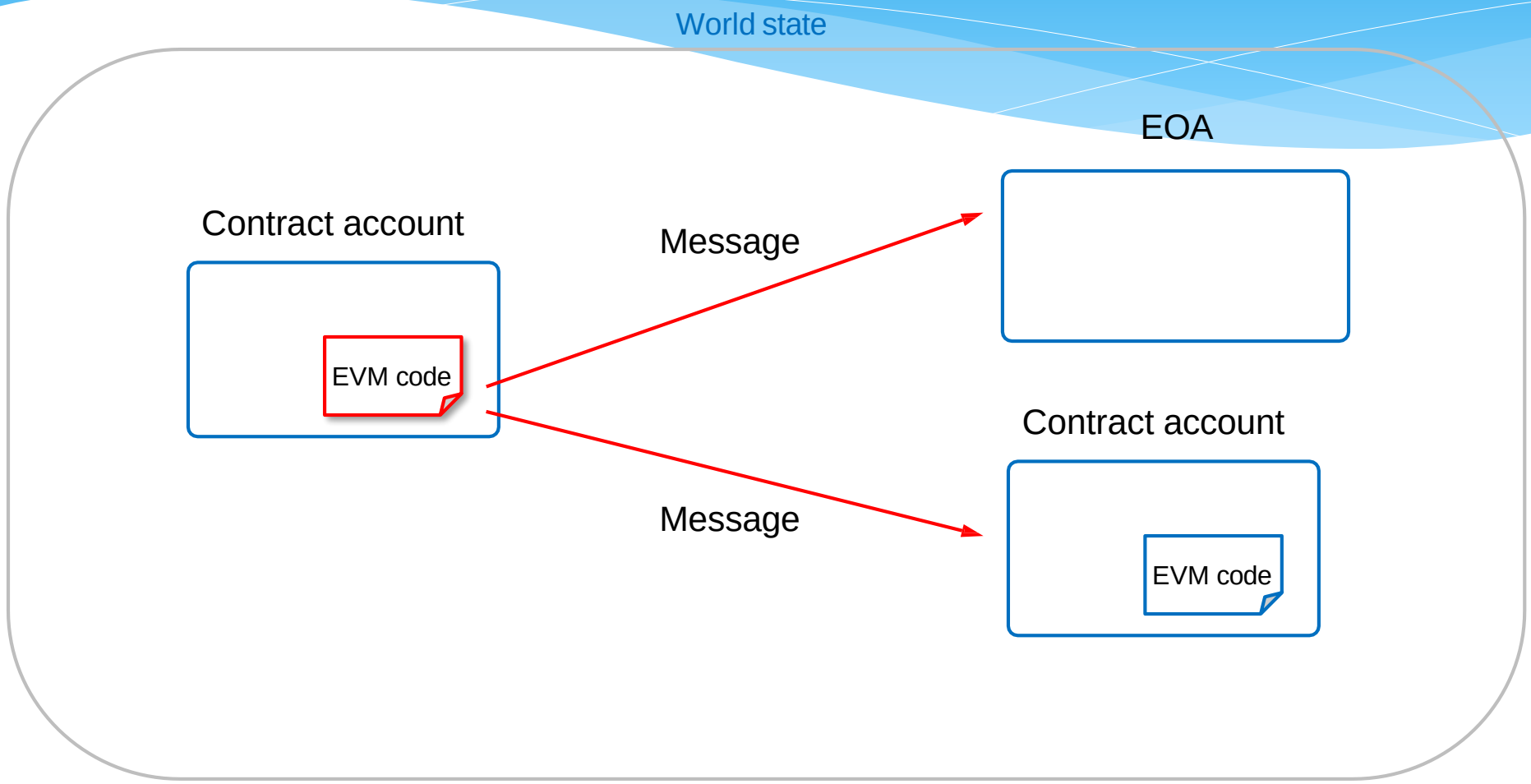
```
PUSH1  e0
PUSH1  02
EXP
PUSH1  00
CALLDATALOAD
    :
```

Bytecode view

```
0x60e060020a600035...
```

EVM Code is the bytecode that the EVM can natively execute.

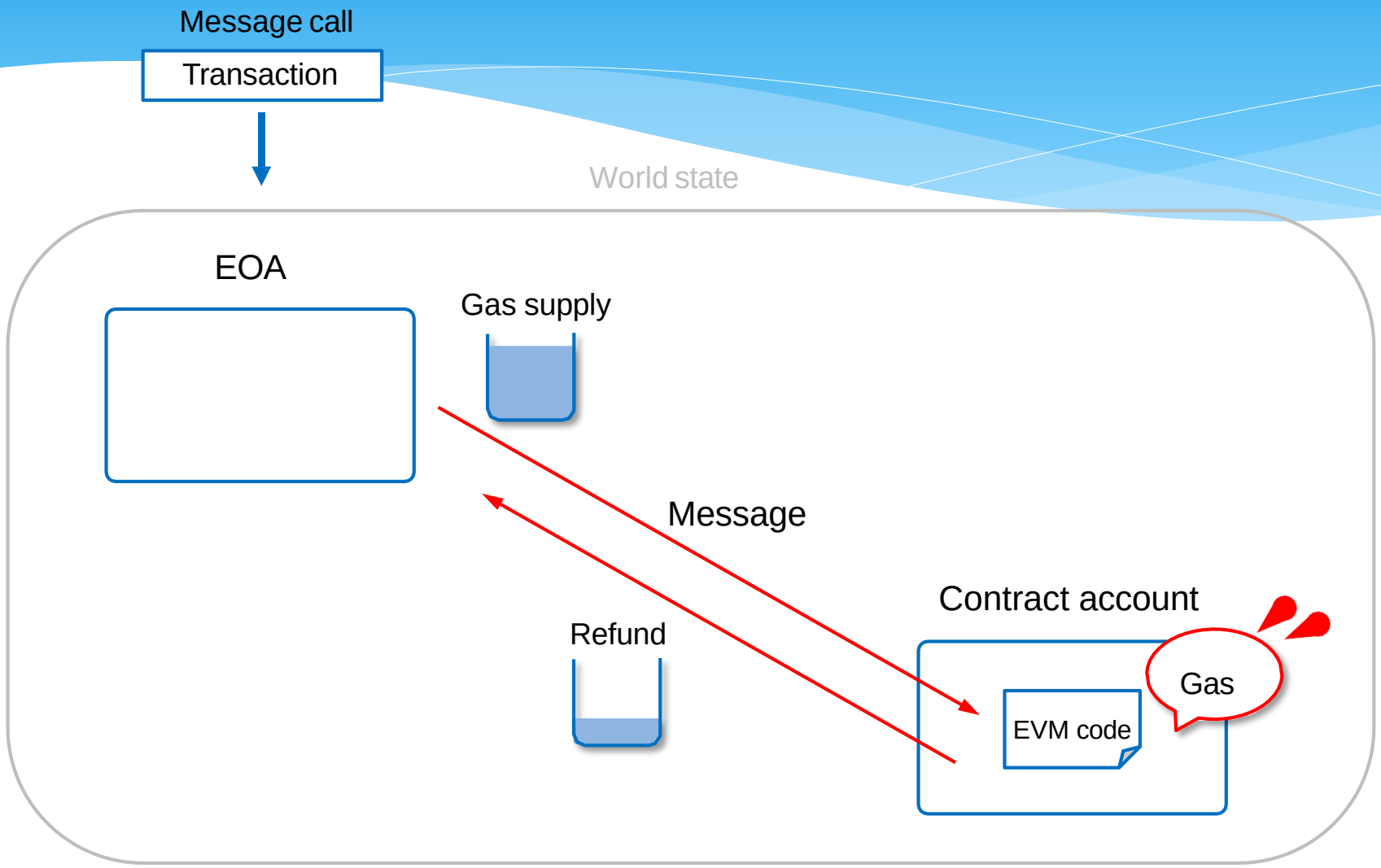
Message call



EVM can send a message to other account.

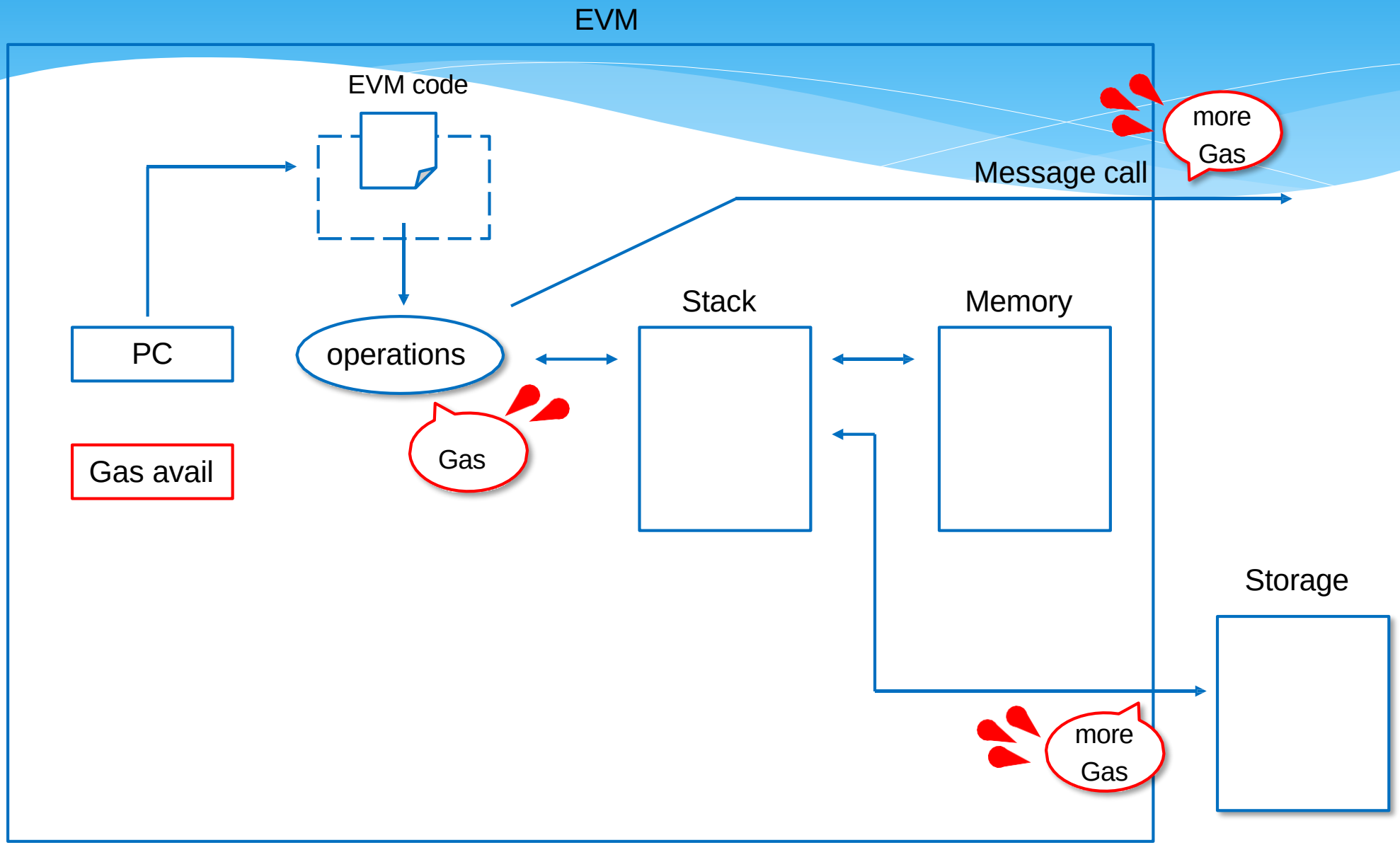
The depth of message call is limited to less than 1024 levels.

Gas and fee

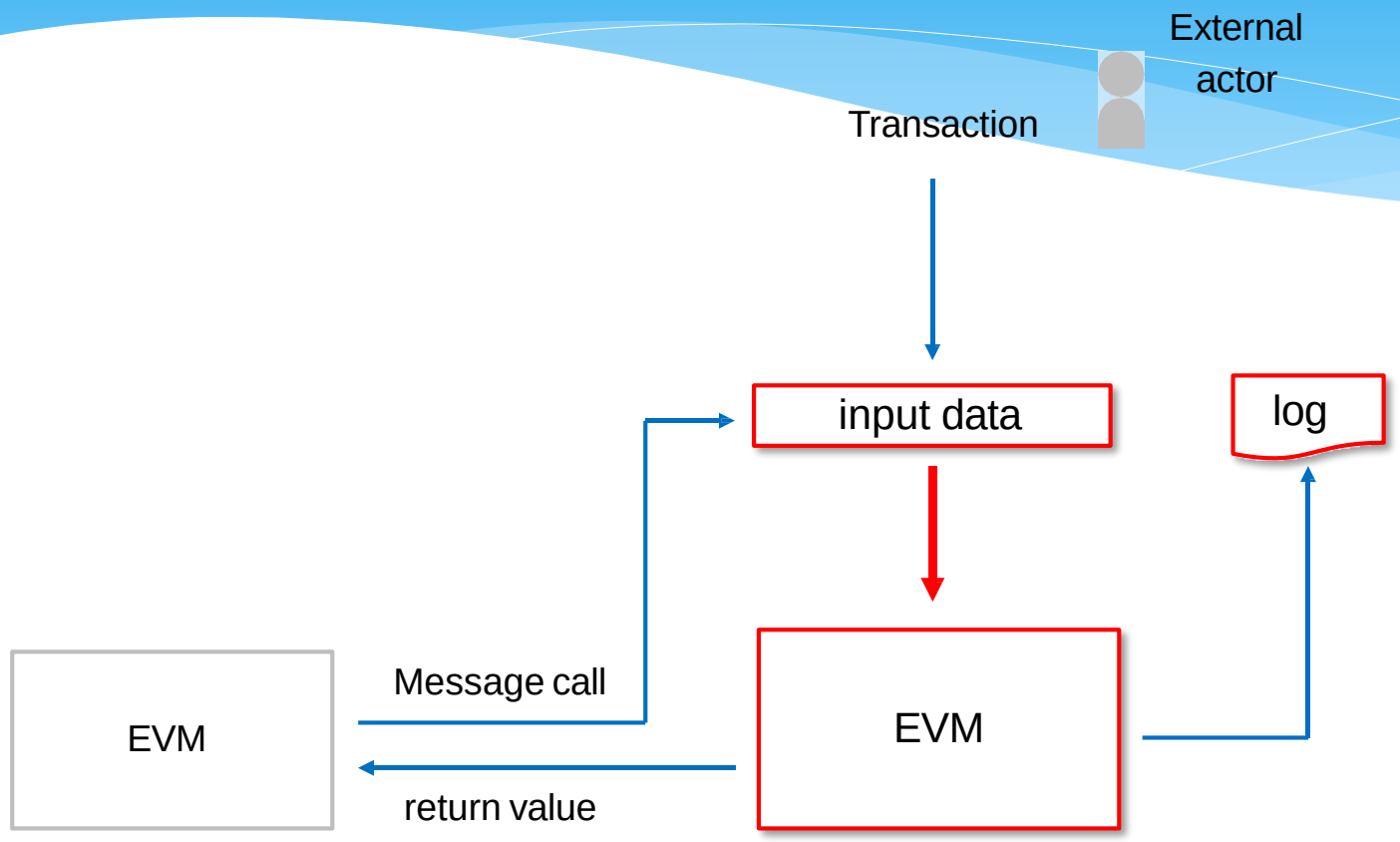


All programmable computation in Ethereum is subject to fees (denominated in gas).

Gas


EVM
EVM code
more
Gas
Message call
Stack
Memory
PC
operations
Gas avail
Gas
Storage
more
Gas

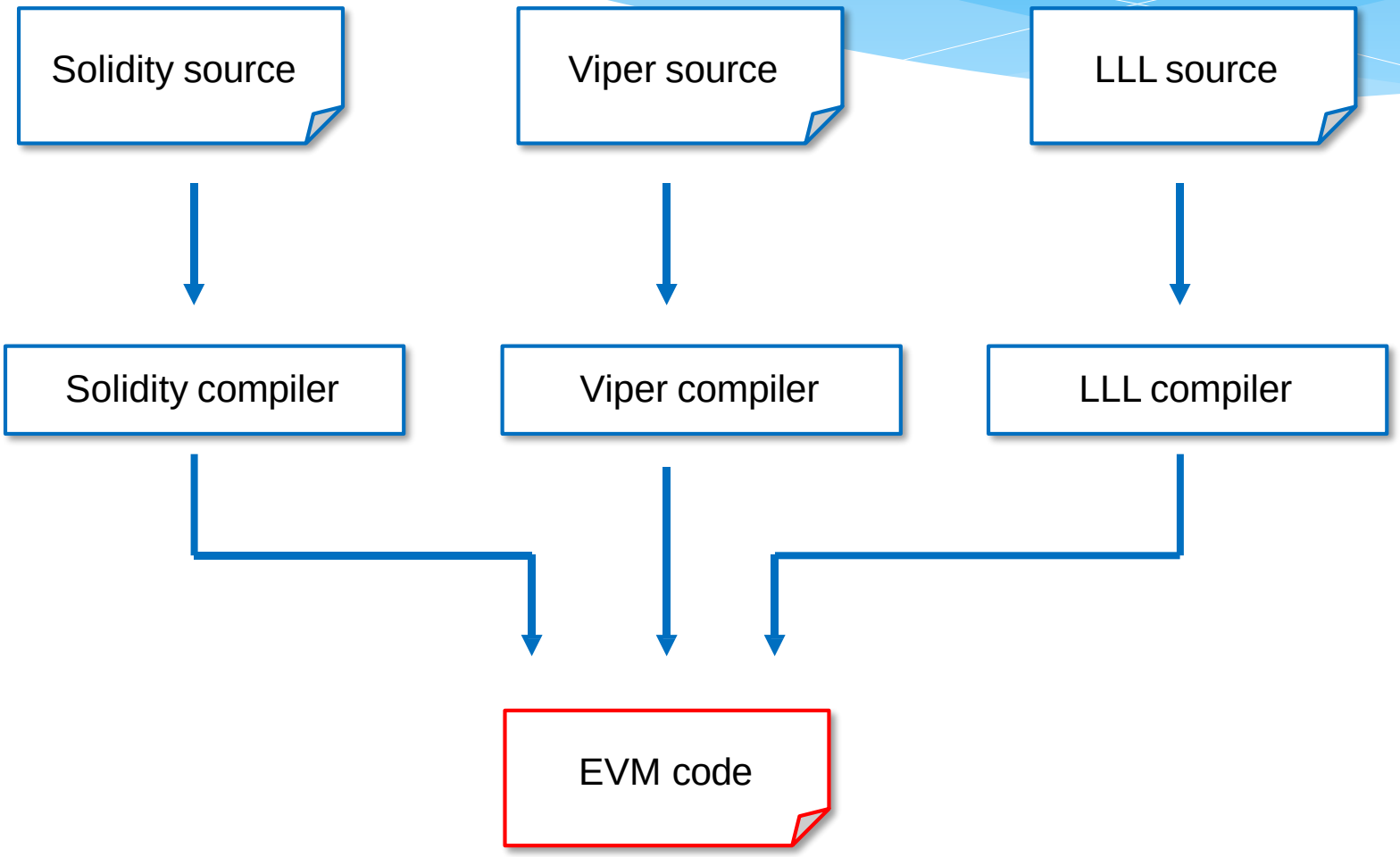
Input and Output of EVM



EVM can input external data from a message call.

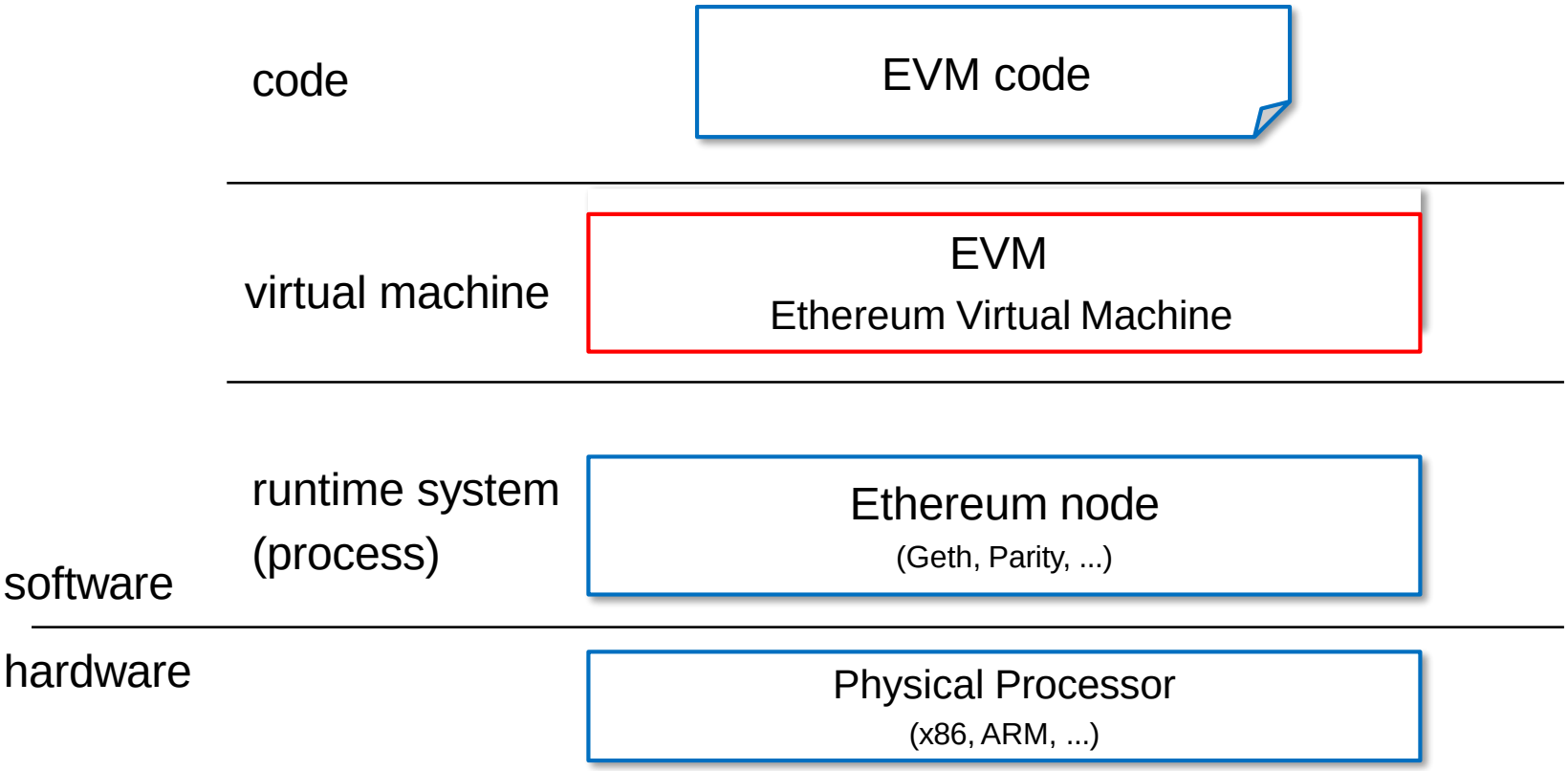
EVM can output log. EVM can also return values to Caller EVM.

EVM code generation



Ethereum virtual machine code

Ethereum virtual machine layer



Solidity

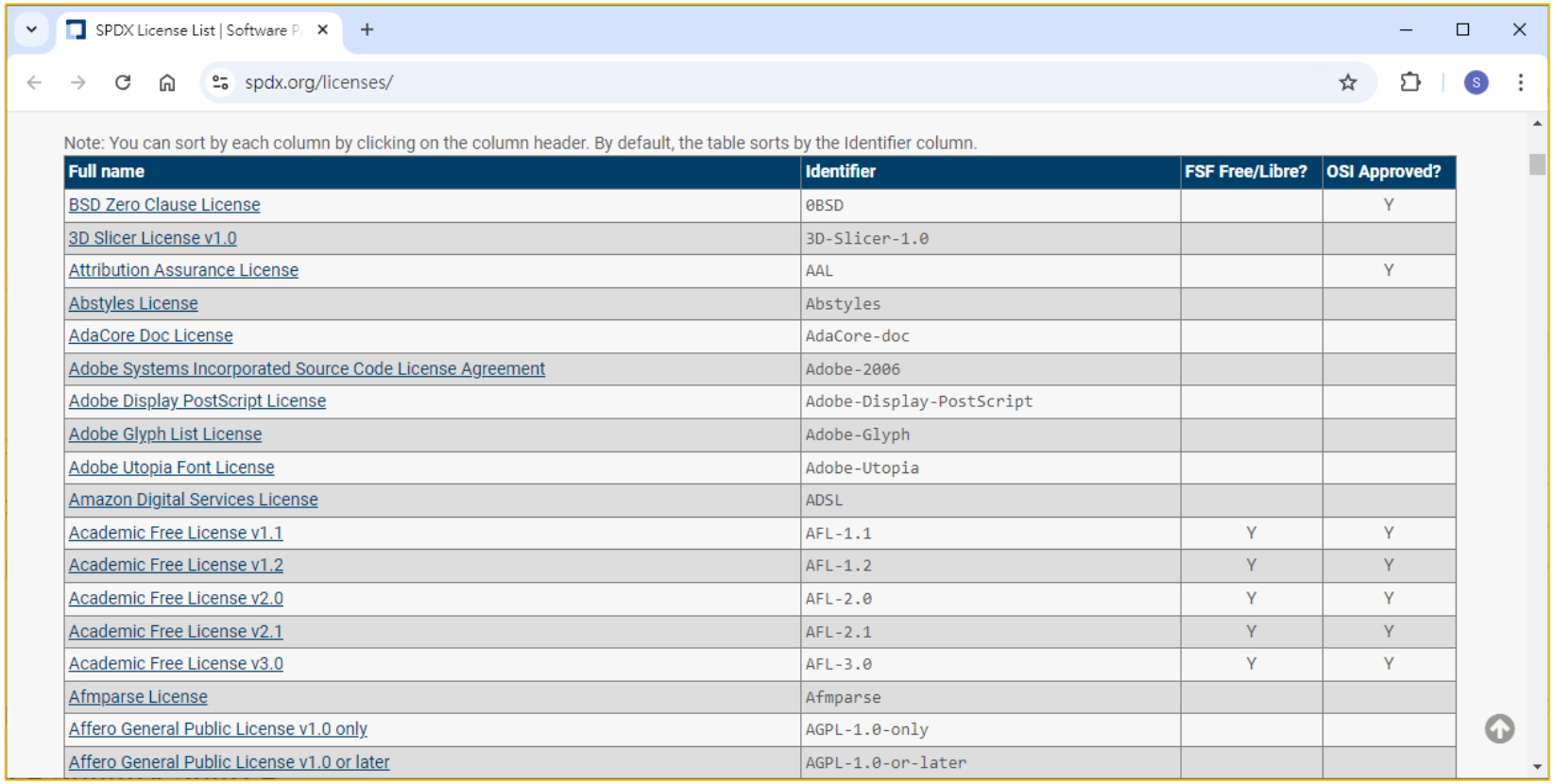
SPDX license識別碼

- SPDX(System Package Data Exchange)是一種開放標準，能夠表示具有SBOM（Software Bill of Materials/軟體物料清單）等軟體元件的系統，以及支援一系列風險管理使用案例的其他人工智慧、資料和安全參考。
- SPDX 規格是提供完全免費的國際開放標準(ISO/IEC 5692:2021)。
- 建立合約前，必須在最前面聲明採用那一個識別碼(identifier)。

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;
```

SPDX license List

- 完整的SPDX清單於 https://spdx.org/licenses/



Note: You can sort by each column by clicking on the column header. By default, the table sorts by the Identifier column.

Full name	Identifier	FSF Free/Libre?	OSI Approved?
BSD Zero Clause License	0BSD		Y
3D Slicer License v1.0	3D-Slicer-1.0		
Attribution Assurance License	AAL		Y
Abstyles License	Abstyles		
AdaCore Doc License	AdaCore-doc		
Adobe Systems Incorporated Source Code License Agreement	Adobe-2006		
Adobe Display PostScript License	Adobe-Display-PostScript		
Adobe Glyph List License	Adobe-Glyph		
Adobe Utopia Font License	Adobe-Utopia		
Amazon Digital Services License	ADSL		
Academic Free License v1.1	AFL-1.1	Y	Y
Academic Free License v1.2	AFL-1.2	Y	Y
Academic Free License v2.0	AFL-2.0	Y	Y
Academic Free License v2.1	AFL-2.1	Y	Y
Academic Free License v3.0	AFL-3.0	Y	Y
Afmparse License	Afmparse		
Affero General Public License v1.0 only	AGPL-1.0-only		
Affero General Public License v1.0 or later	AGPL-1.0-or-later		

備註說明

- 單行備註使用雙斜線//在說明的前面。
- 多行備註使用 (/*...*/) 把備註包括在裏面。

```
// This is a single-line comment.

/*
This is a
multi-line comment.
*/
```

版本宣告

- 每次建立合約前，必須聲明solidity採用的版本。

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

//pragma solidity >=0.8.4  <0.9.0;
//Demo updated build Aug/10/2024
contract MyDemo {
    // Basic variables
    address _owner;
 }
```

- 程式碼格式類以js，以分號;來區隔指令。
- ^0.8.4表示compiler只適用於0.8.4<=X<0.9.0之間。

Importing other Source Files

- 使用import來載入其他的.sol檔案。

```
import "filename";

// import '@openzeppelin/contracts/token/ERC721/IERC721.sol';
```

- 建立一個新的全域符號symbolName，其成員是「filename」中的所有全域符號。

```
import * as symbolName from "filename";

// import {IERC20} from "./IERC20.sol";
```

狀態變數(State Variables)

- 狀態變數會被永久存入合約中，類似於寫入database。

```
// 宣告變數的方式，也與 js 非常類似
// contract建立一個合約外層，並在合約內建立變數
 contract MyDemo {
        // 建立一個變數 amount，指定其為uint資料型態，並給予賦值
        uint amount = 10;   // State variable
}
```

加減運算

- solidity的數學計算，明確直白，就是一般數學符號進行計算。

```
1 + 1
2 - 1
10 * 2
6 / 3
2 ** 2  (平方運算)
因此我們也可以宣告變數來當作運算用數字

contract MyDemo {
        uint qty = 5;
        uint price = 10;
        uint totalPrice = qty * price;
}
```

String Literals and Types

- String literals can only contain printable ASCII characters, which means the characters between and including 0x20 ~ 0x7E.
- 另支援以下特殊字元。

- `\<newline>` (escapes an actual newline)
- `\\` (backslash)
- `\'` (single quote)
- `\"` (double quote)
- `\n` (newline)
- `\r` (carriage return)
- `\t` (tab)
- `\xNN` (hex escape)
- `\uNNNN` (unicode escape)

```
// 宣告變數的方式如下
 string tokenName;
```

Unicode Literals

- While regular string literals can only contain ASCII, Unicode literals – prefixed with the keyword unicode – can contain any valid UTF-8 sequence. They also support the very same escape sequences as regular string literals.

```
// 宣告變數的方式如下

string memory test = unicode"Hello 😃";
```

Address Literals

- Hexadecimal literals that are between 39 and 41 digits long and do not pass the checksum test produce an error.
- You can prepend (for integer types) or append (for bytesNN types) zeros to remove the error.

```
// 宣告變數的方式如下

address  fromAddress = 0xdCad3a6d3569DF655070DEd06cb7A1b2Ccd1D3AF;
```

結構體(struct Types)

- solidity提供一種名為結構體(struct)的資料型態表示方法。
- 結構體允許生成複雜的資料型態，同時擁有多個屬性。
- 下列程式碼中，結構體內的uint、string為資料型態。

```
// 宣告變數的方式如下

 struct Student {   // struct
            string name;
            uint age;
}
```

陣列資料型態(Array)

- solidity有兩種陣列資料型態，分別是靜態與動態。
- 所有陣列都由連續的記憶體位置組成。最低地址對應於第一個元素，最高地址對應於最後一個元素。

```
// 靜態
uint[10] fixedArray; //固定長度為10的靜態數字陣列
string[20] stringArray; //固定長度為20且是string類型的靜態陣列
uint balance[3] = [1, 2, 3]; //固定長度為3,且設定初始值
uint balance[] = [1, 2, 3];   //和上面的作法是一樣的

// 動態
uint[] dynamicArray; //長度不固定的動態數字陣列
```

- 公開陣列，提供保存公開資料，僅供其他合約讀取資料，而不能寫入。

```
// 使用public宣告公開陣列

myAccount[] public my;
```

Function

- Functions are the executable units of code. Functions are usually defined inside a contract, but they can also be defined outside of contracts.

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.1 <0.9.0;

contract SimpleAuction {
    function bid() public payable { // Function
        // ...
    }
}

// Helper function defined outside of a contract
function helper(uint x) pure returns (uint) {
    return x * 2;
}
```

Events

- Events are convenience interfaces with the EVM logging facilities。

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.22;

event HighestBidIncreased(address bidder, uint amount); // Event

contract SimpleAuction {
    function bid() public payable {
        // ...
        emit HighestBidIncreased(msg.sender, msg.value); // Triggering event
    }
}
```

revert

- A direct revert can be triggered using the revert statement and the revert function.
- The revert statement takes a custom error as direct argument without parentheses:

  revert CustomError(arg1, arg2);
- For backward-compatibility reasons, there is also the revert() function, which uses parentheses and accepts a string:

  revert(); revert(“description”);
- The error data will be passed back to the caller and can be caught there. Using revert() causes a revert without any error data while revert("description") will create an Error(string) error.

Errors

- Errors allow you to define descriptive names and data for failure situations. Errors can be used in revert statements. In comparison to string descriptions, errors are much cheaper and allow you to encode additional data. You can use NatSpec to describe the error to the user.

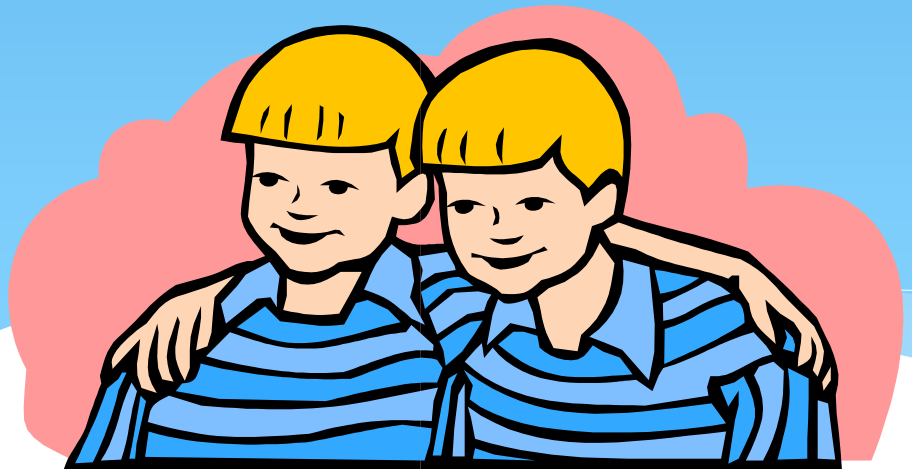
```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.4;

/// Not enough funds for transfer. Requested `requested`,
/// but only `available` available.
error NotEnoughFunds(uint requested, uint available);

contract Token {
    mapping(address => uint) balances;
    function transfer(address to, uint amount) public {
        uint balance = balances[msg.sender];
        if (balance < amount)
            revert NotEnoughFunds(amount, balance);
            revert("Not enough Ether provided.");
        balances[msg.sender] -= amount;
        balances[to] += amount;
        // ...
    }
}
```

Enum Types

- Enums can be used to create custom types with a finite set of ‘constant values’ (see Enums in types section).

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.0 <0.9.0;

contract Purchase {
    enum State { Created, Locked, Inactive } // Enum
}
```

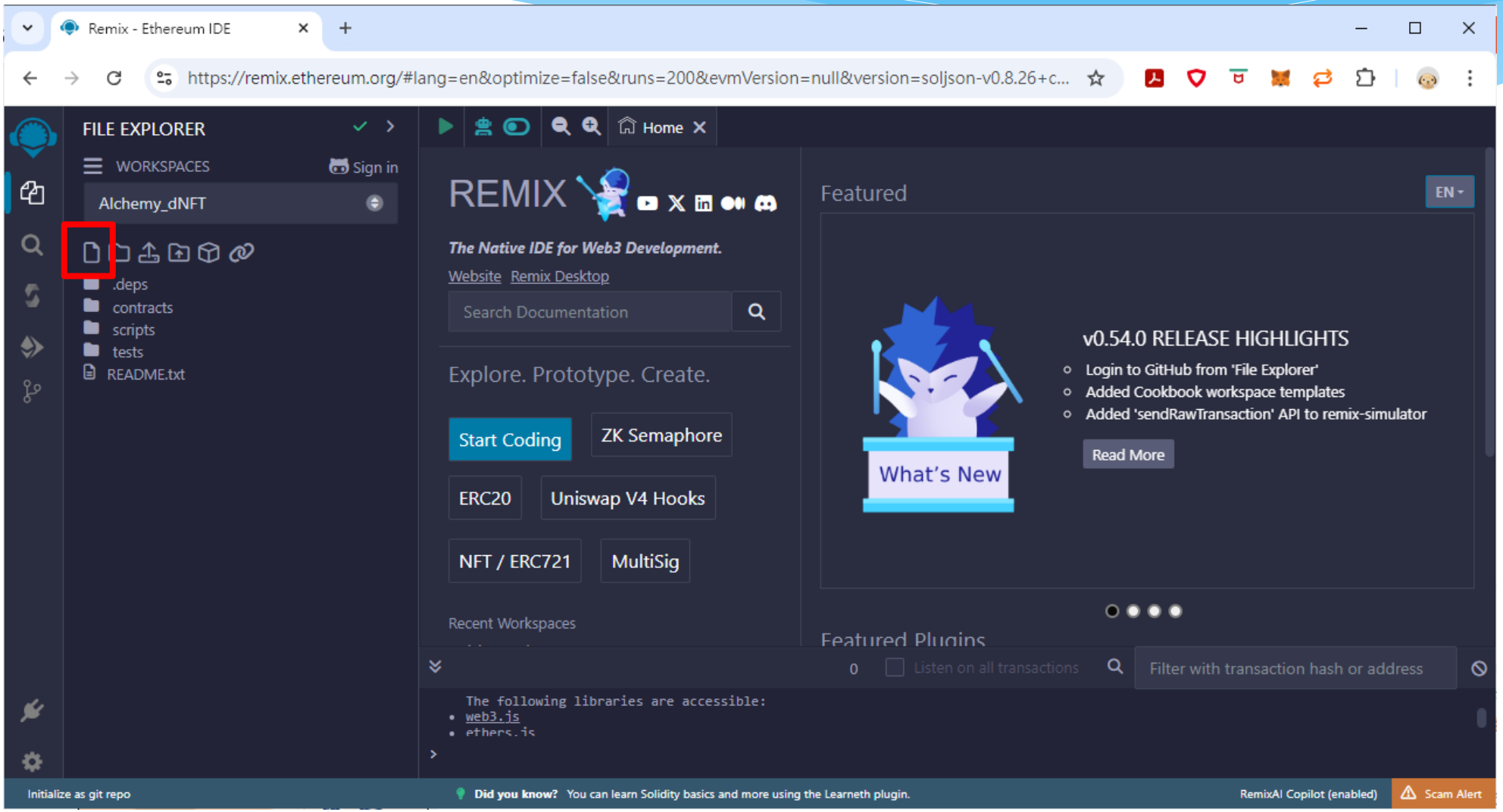
Thank you ！



Start to Deploy Smart Contract

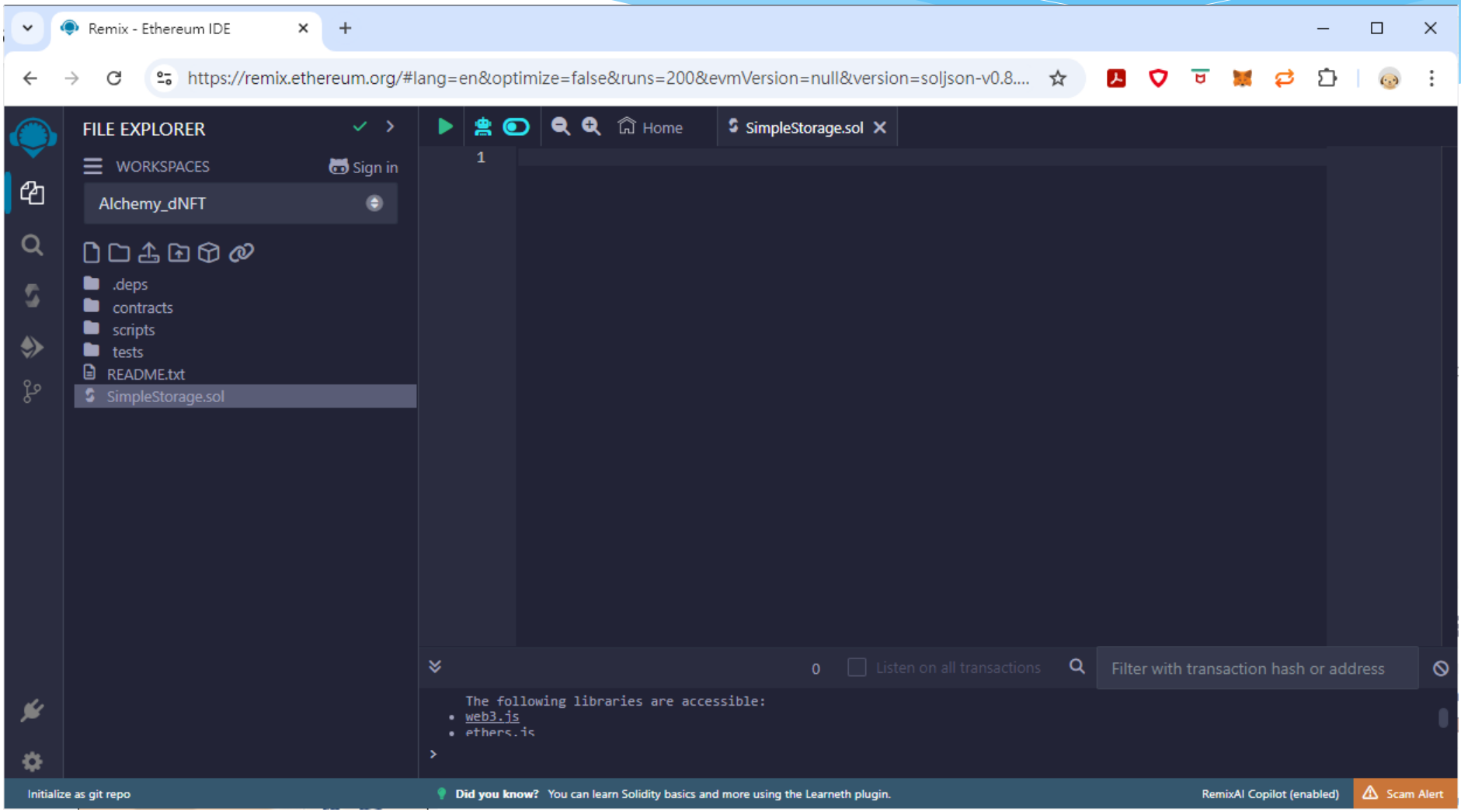
1. Deploy Smart Contract to Sepolia Testnet

Create new file

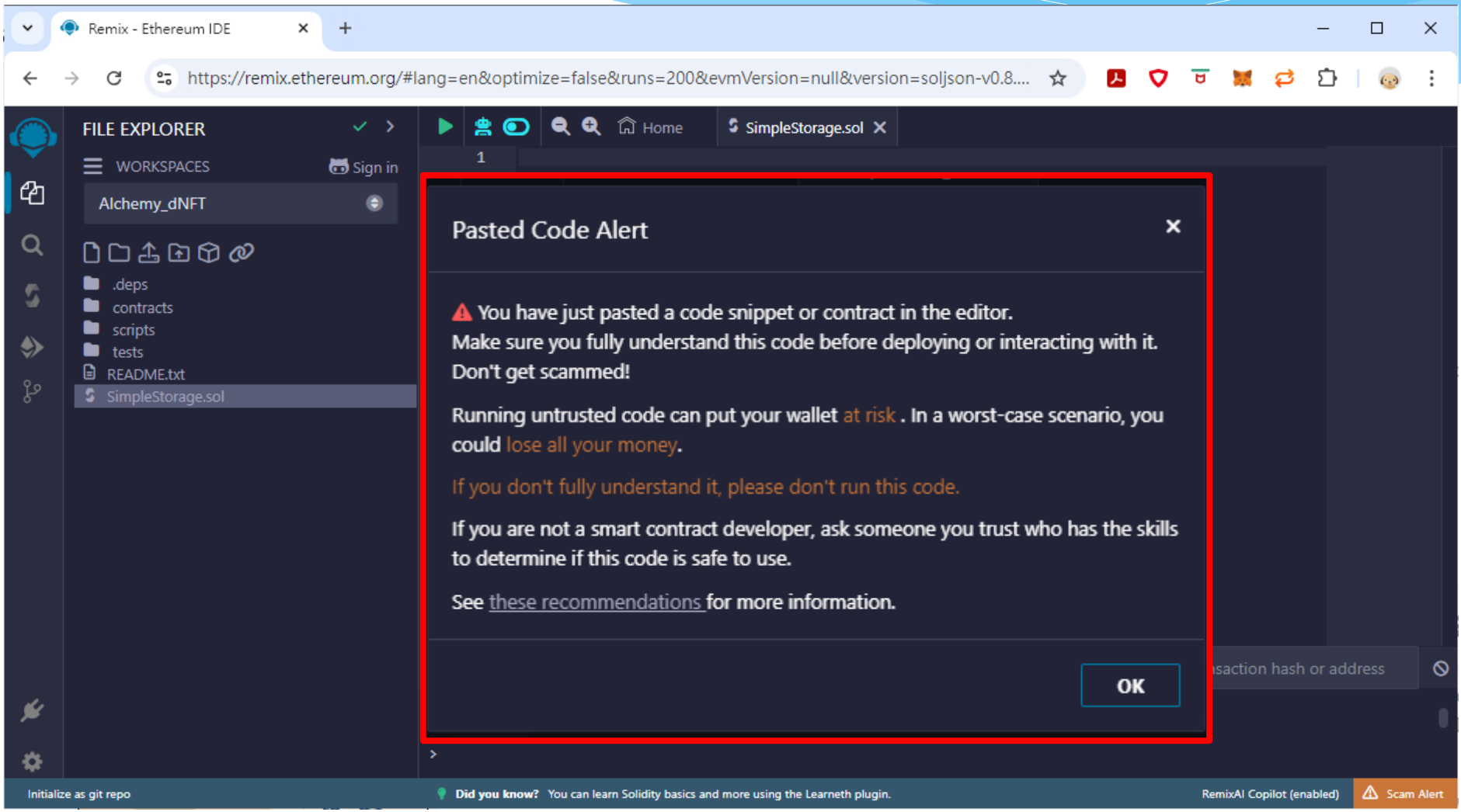


https://remix.ethereum.org

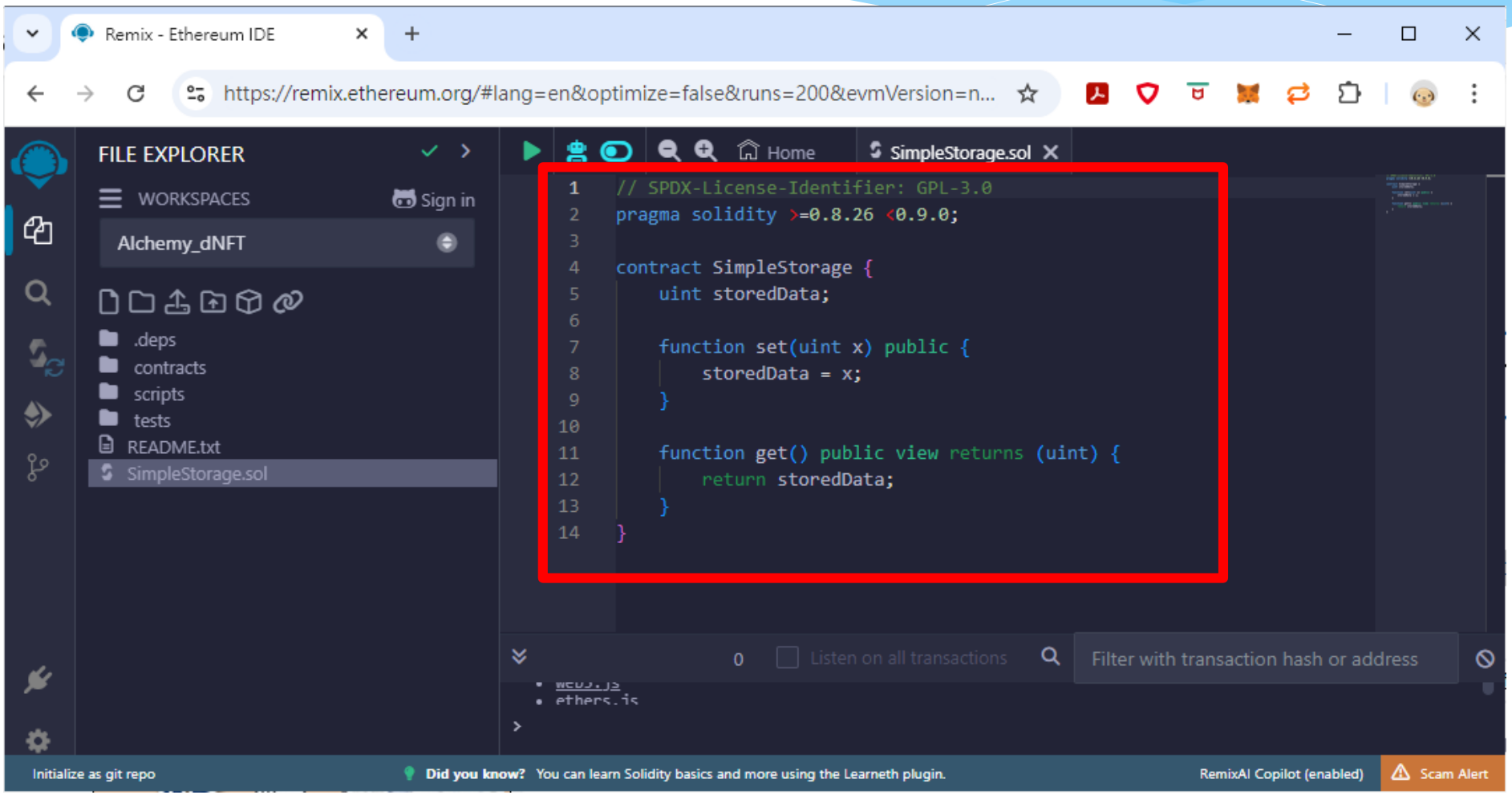
SimpleStorage.sol



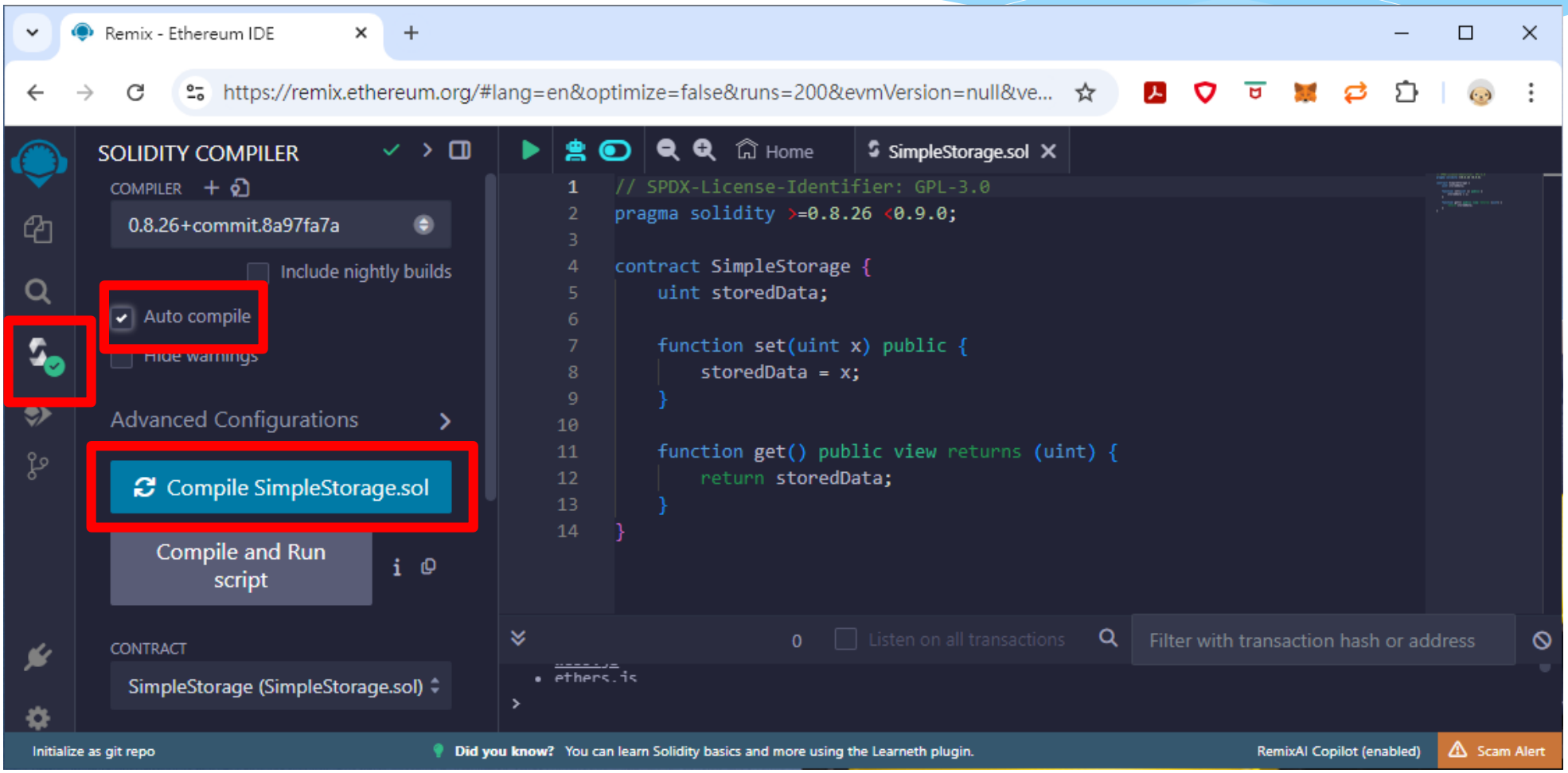
SimpleStorage.sol



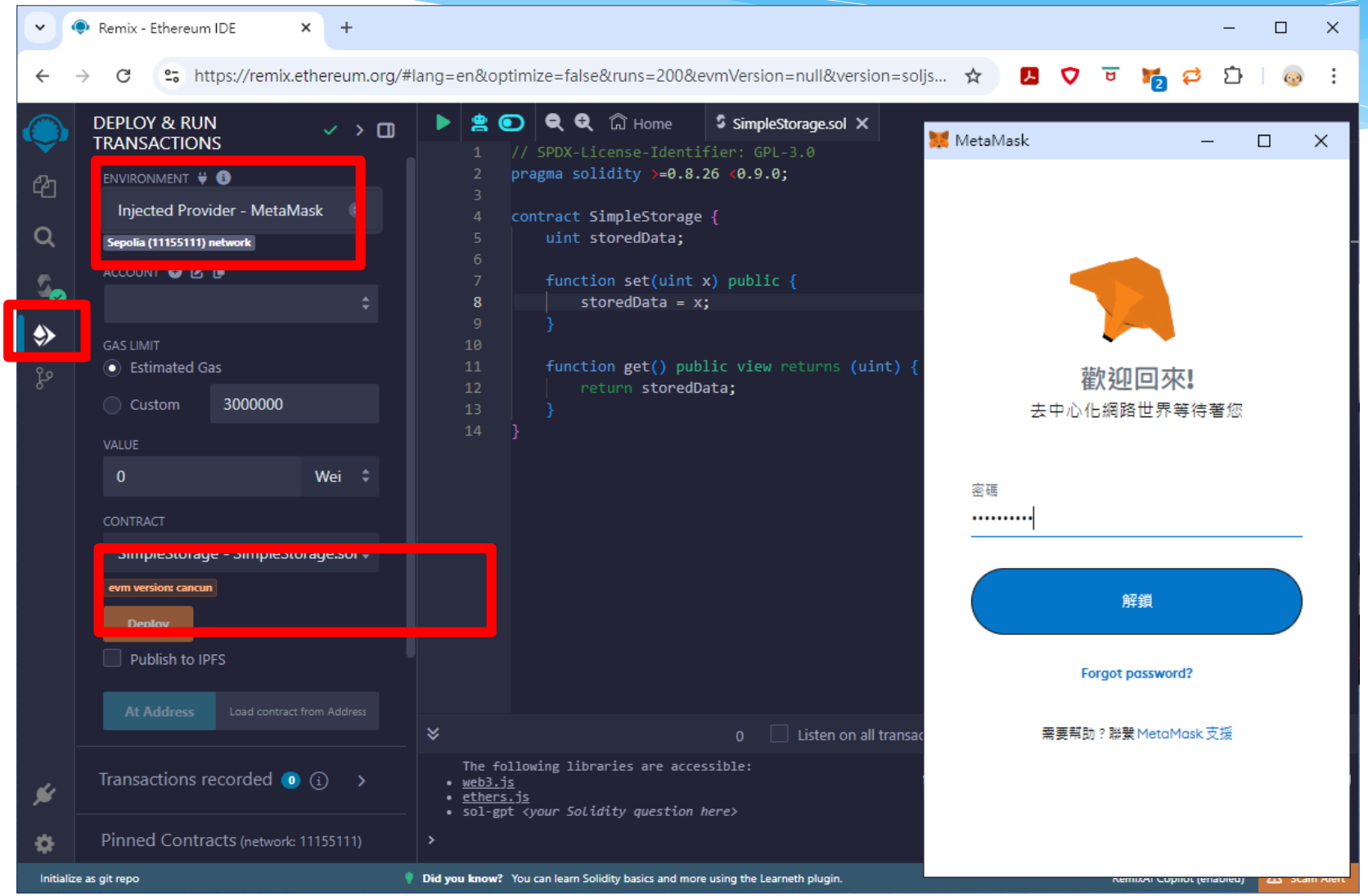
SimpleStorage.sol



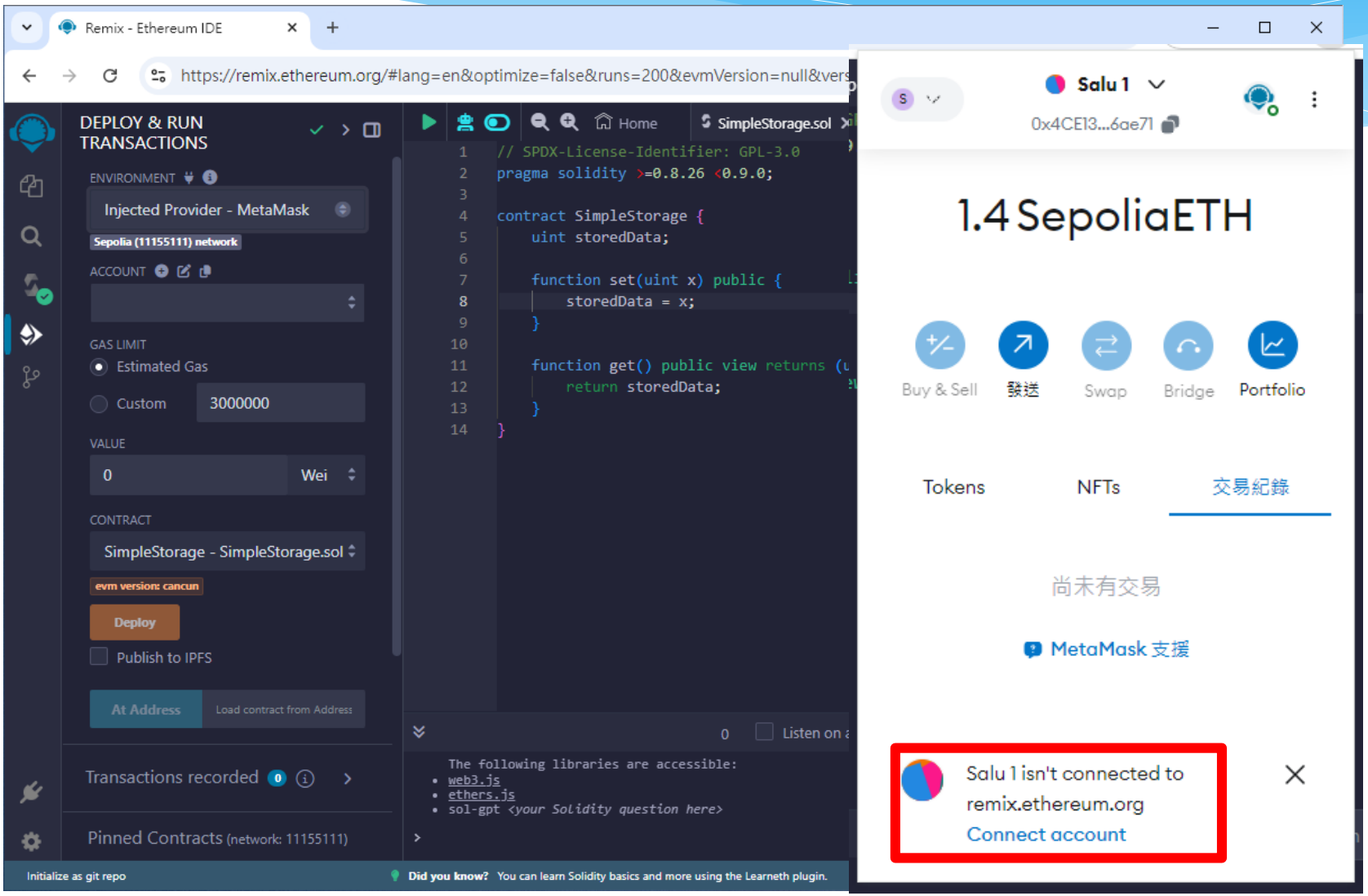
Compile Smart Contract



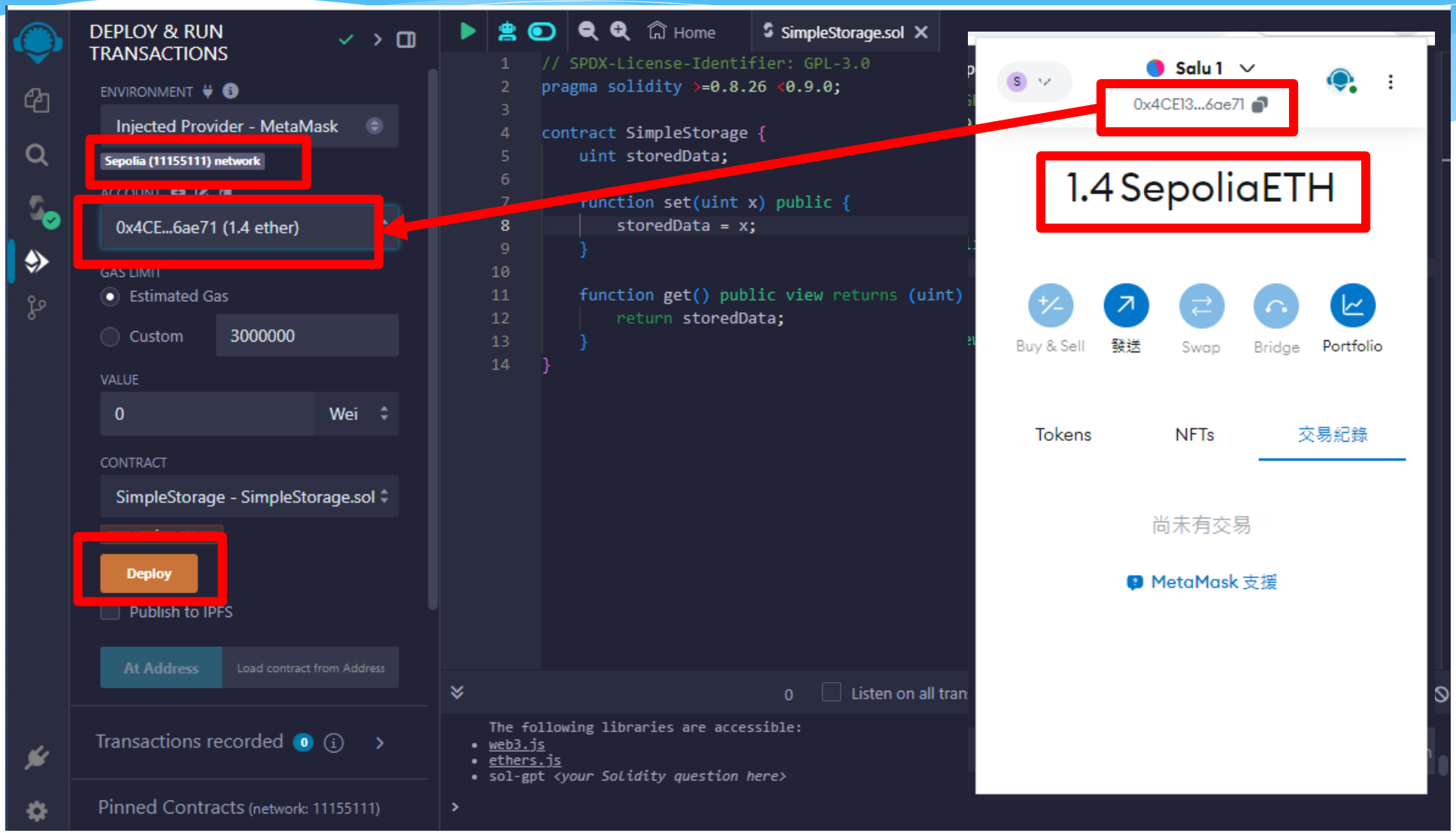
Deploy and Run Transaction



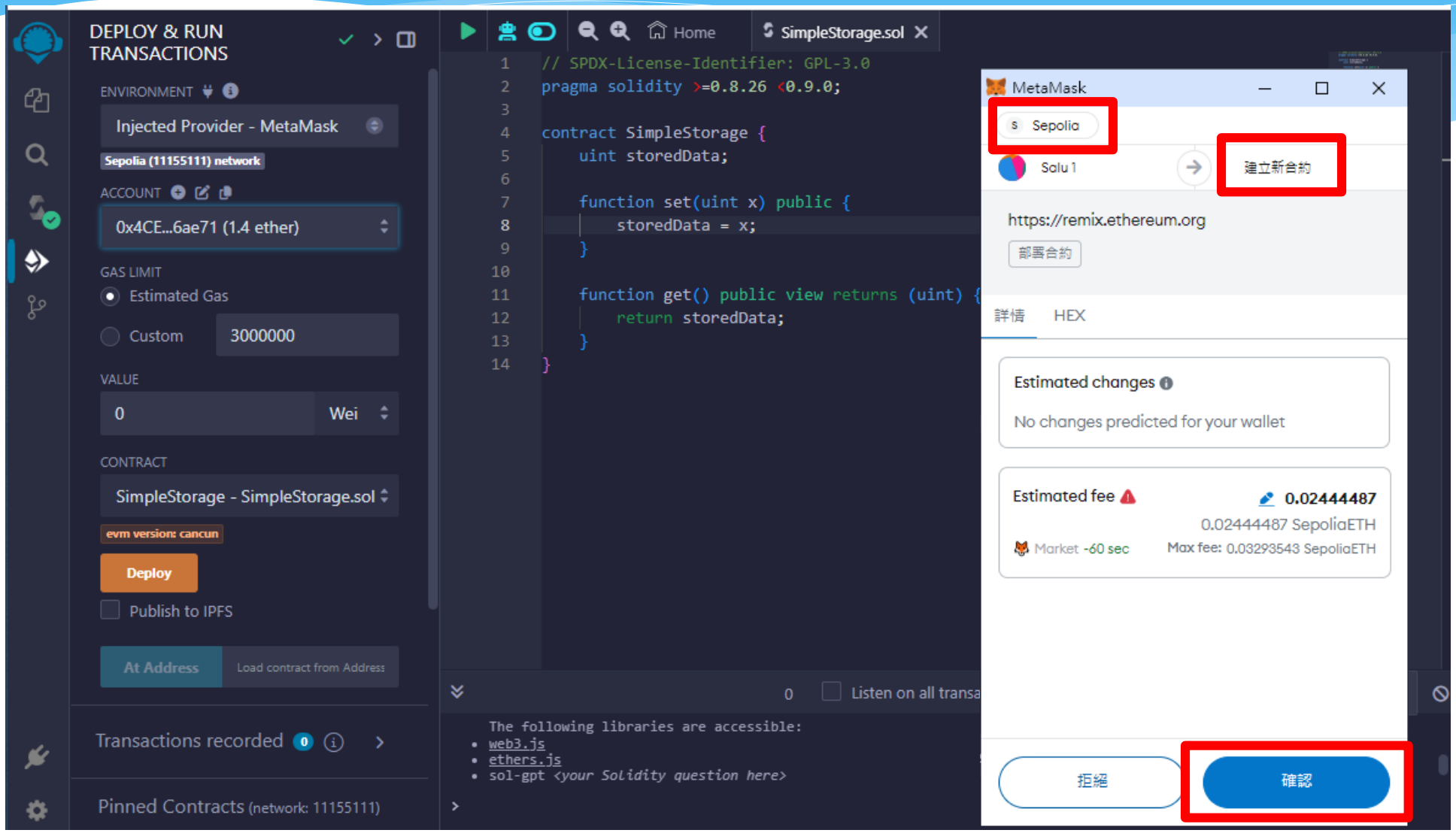
Deploy and Run Transaction



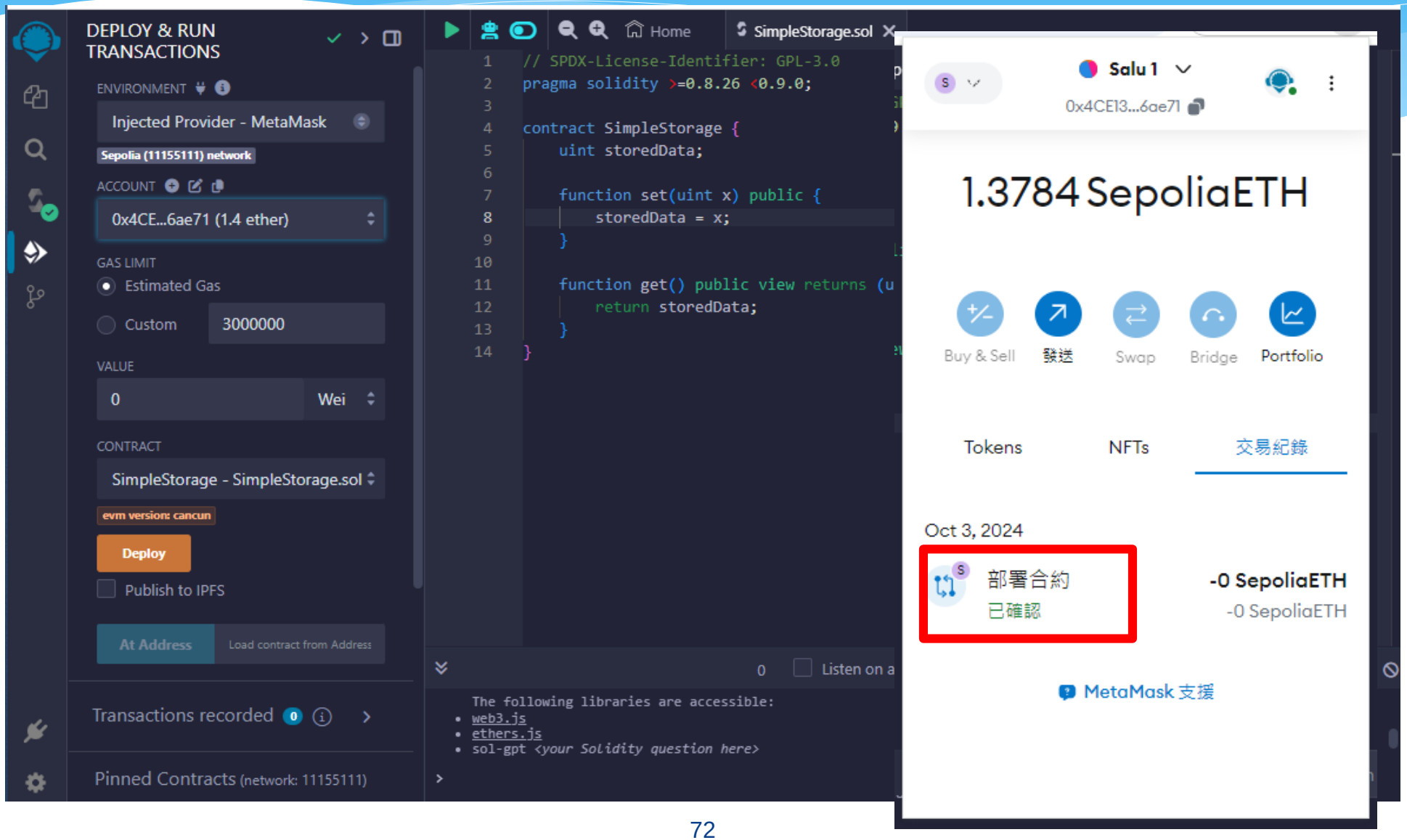
Deploy and Run Transaction



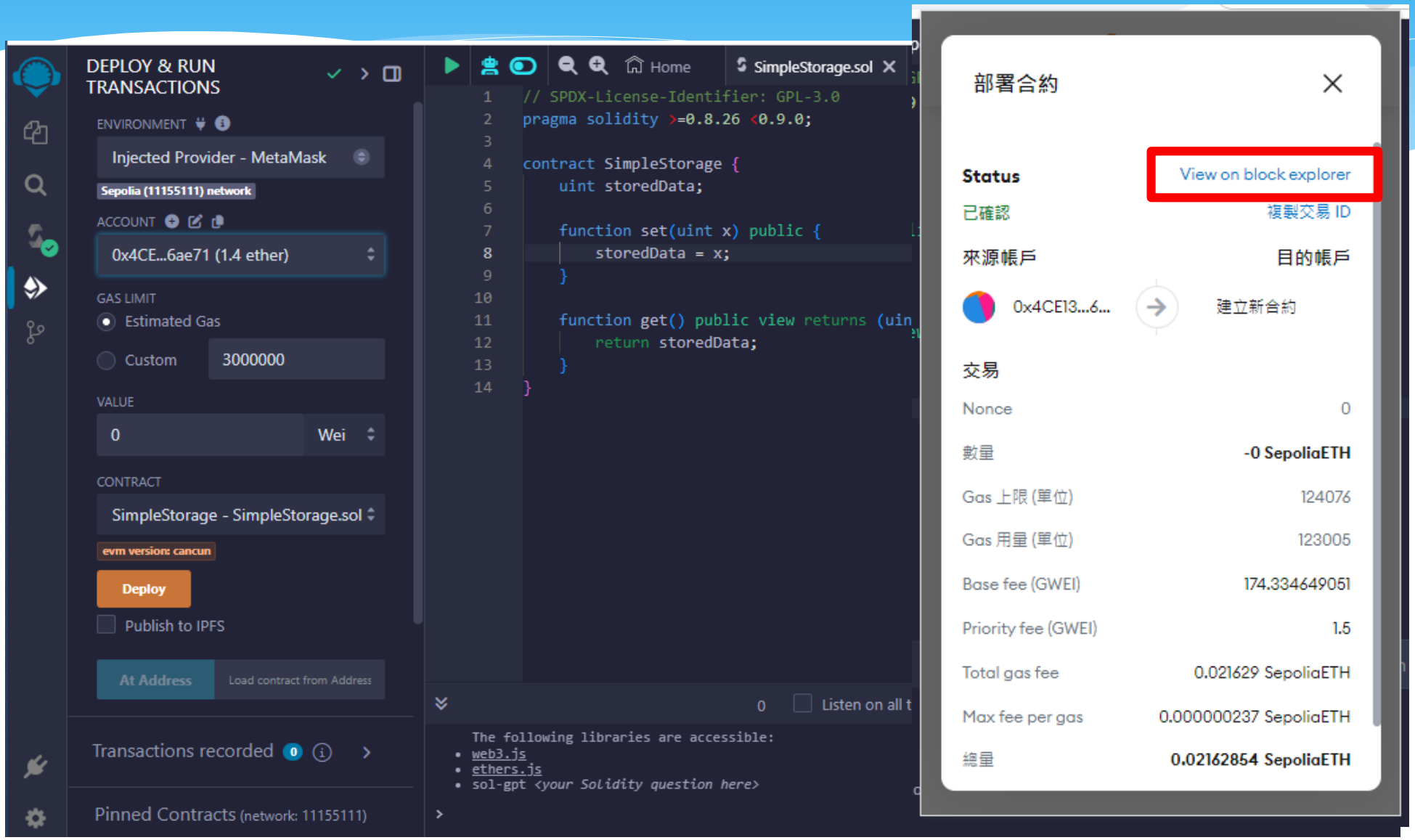
Deploy and Run Transaction



Deploy and Run Transaction

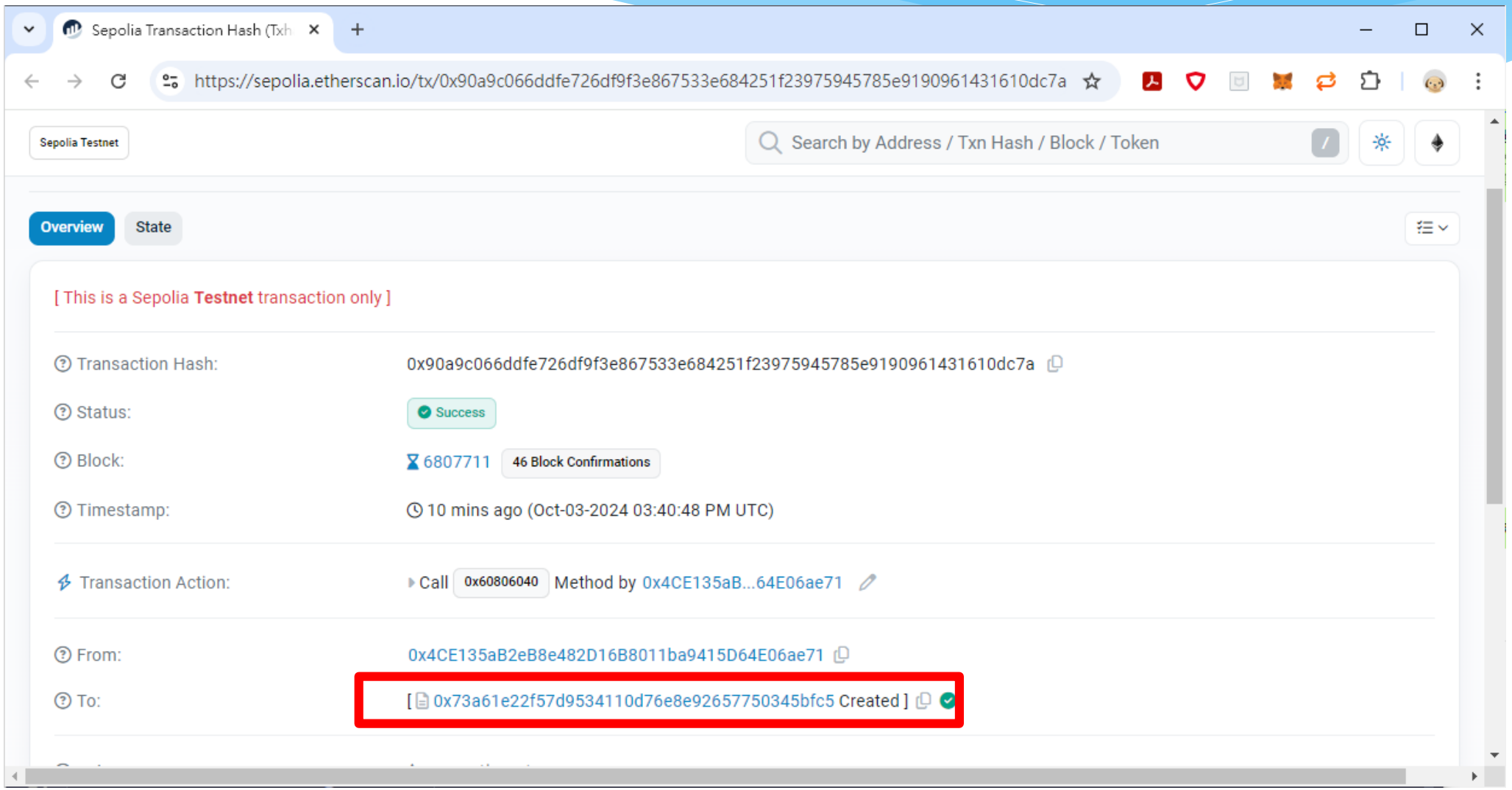


Deploy and Run Transaction



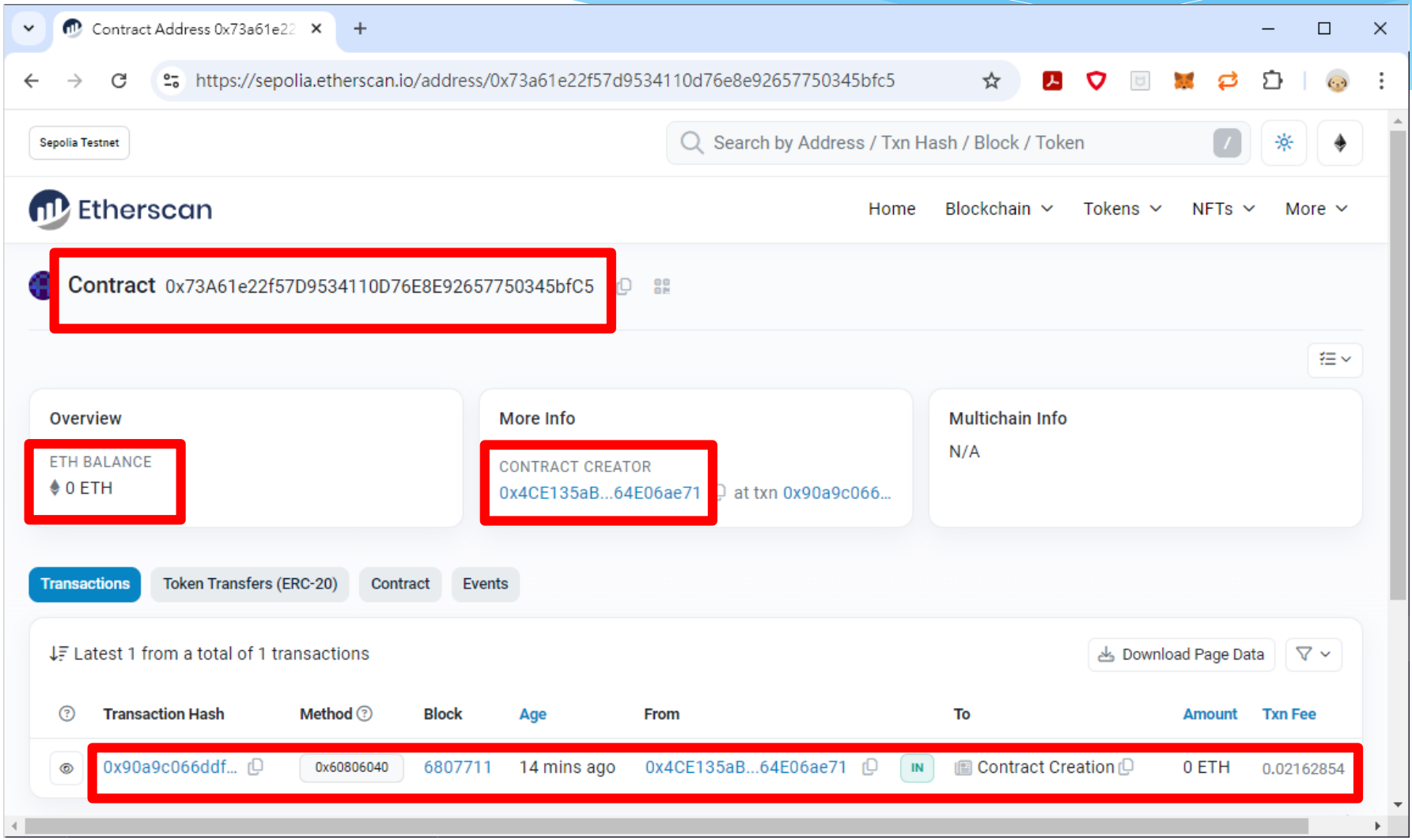
https://sepolia.etherscan.io/tx/0x90a9c066ddfe726df9f3e867533e684251f23975945785e9190961431610dc7a

View Transaction on Sepolia



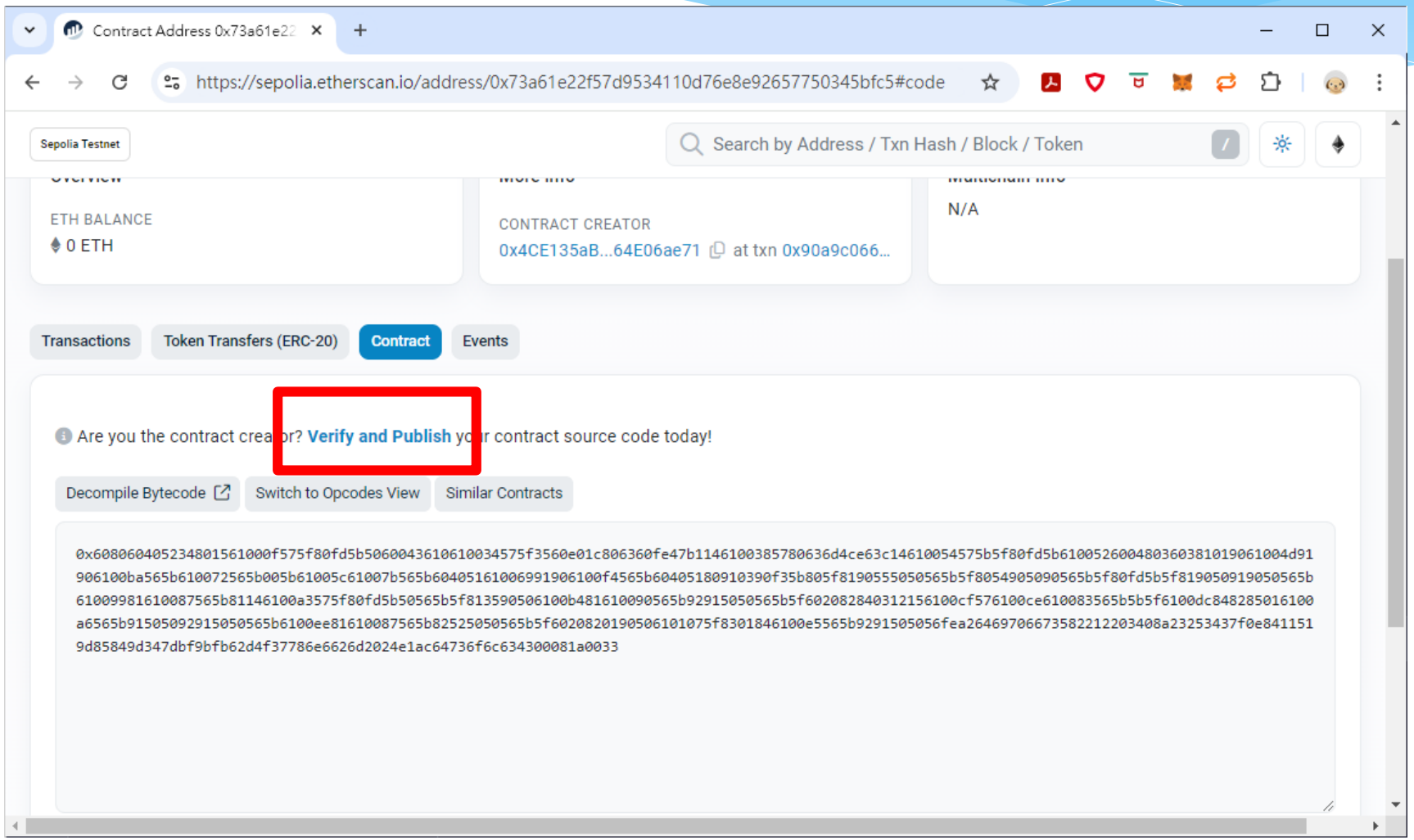
https://sepolia.etherscan.io/tx/0x90a9c066ddfe726df9f3e867533e684251f23975945785e9190961431610dc7a

Smart Contract Information



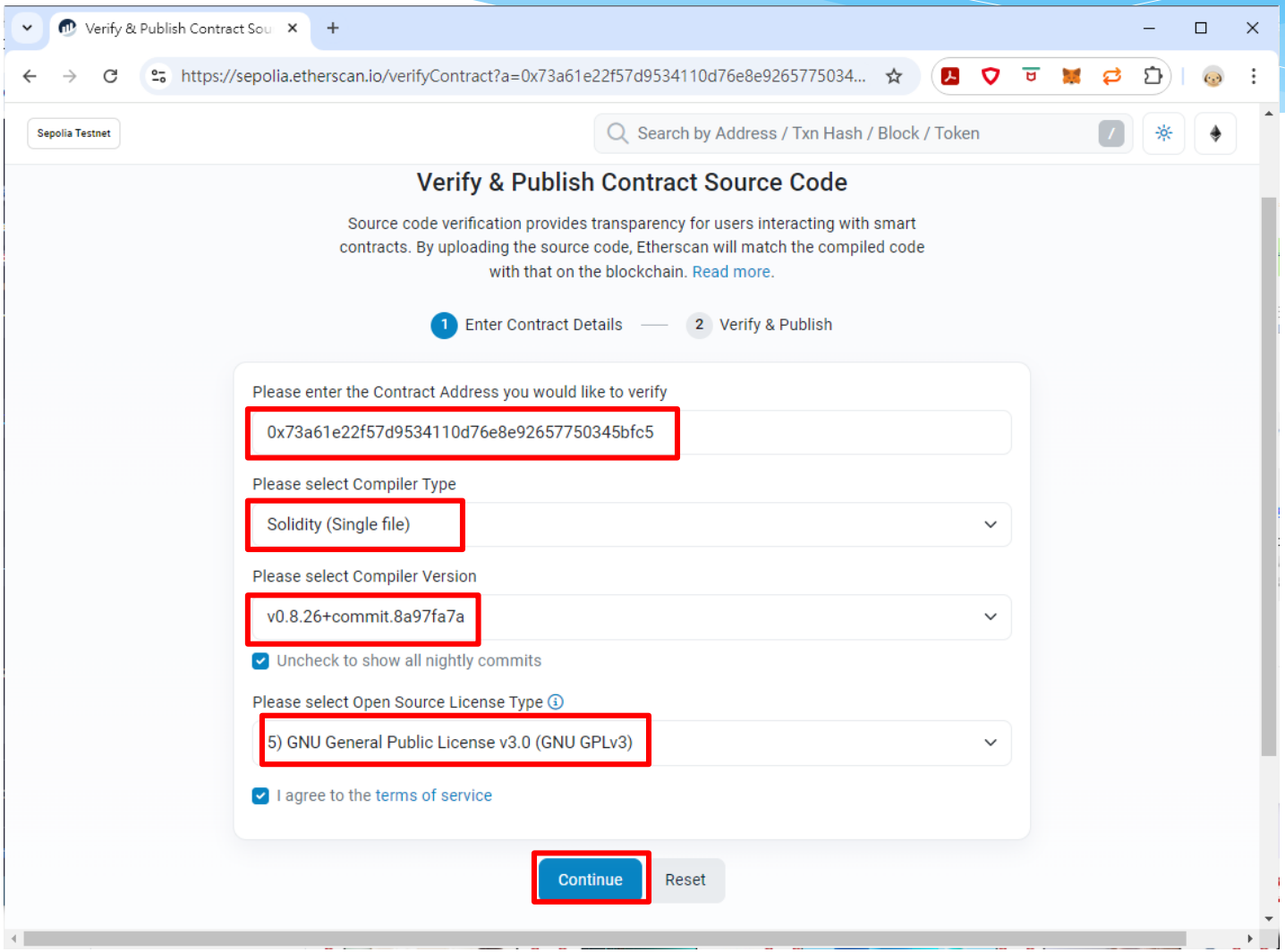
https://sepolia.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5

View & Publish



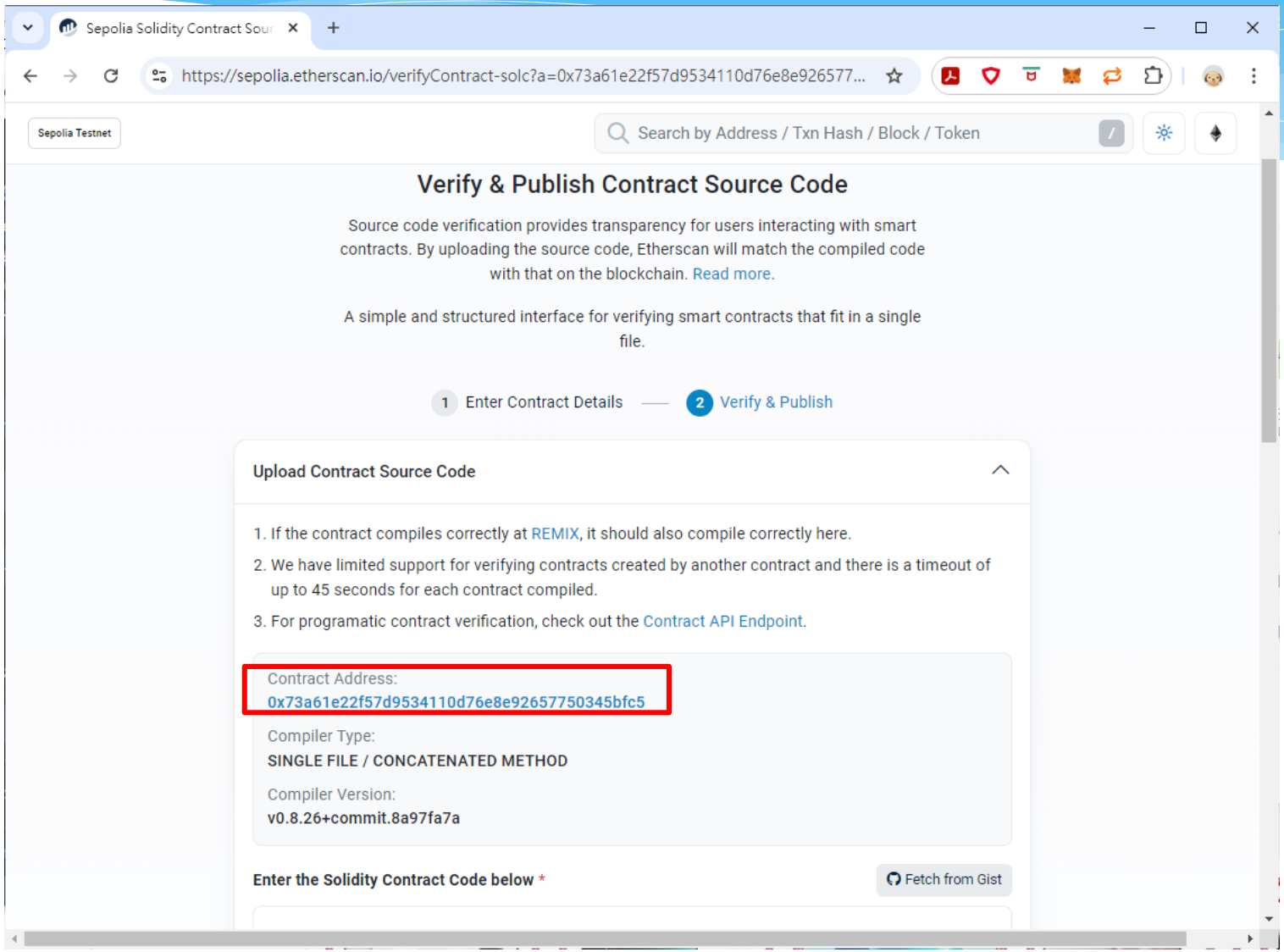
https://sepolia.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5#code

View & Publish Contract Source Code

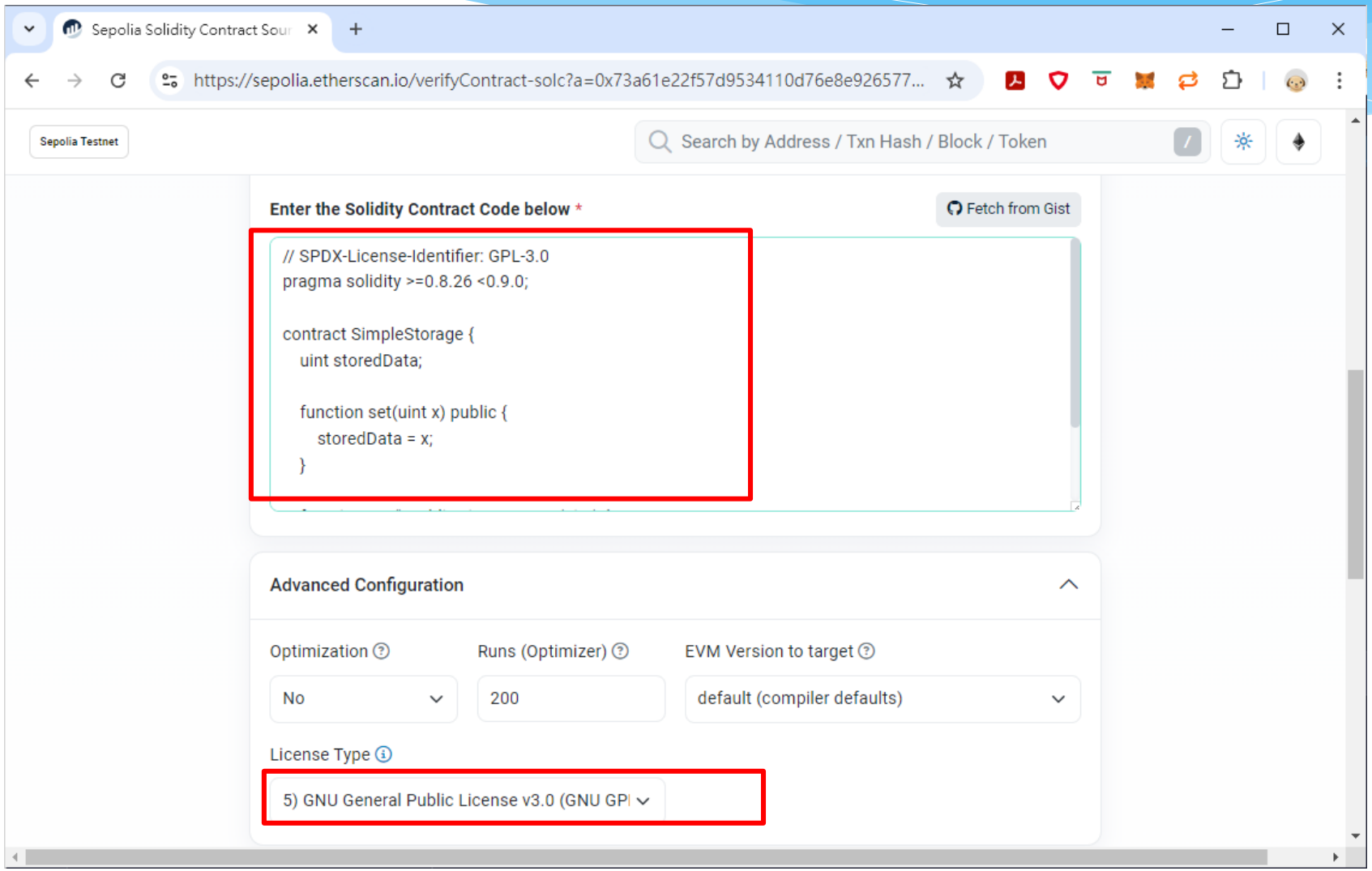


https://sepolia.etherscan.io/verifyContract?a=0x73a61e22f57d9534110d76e8e92657750345bfc5

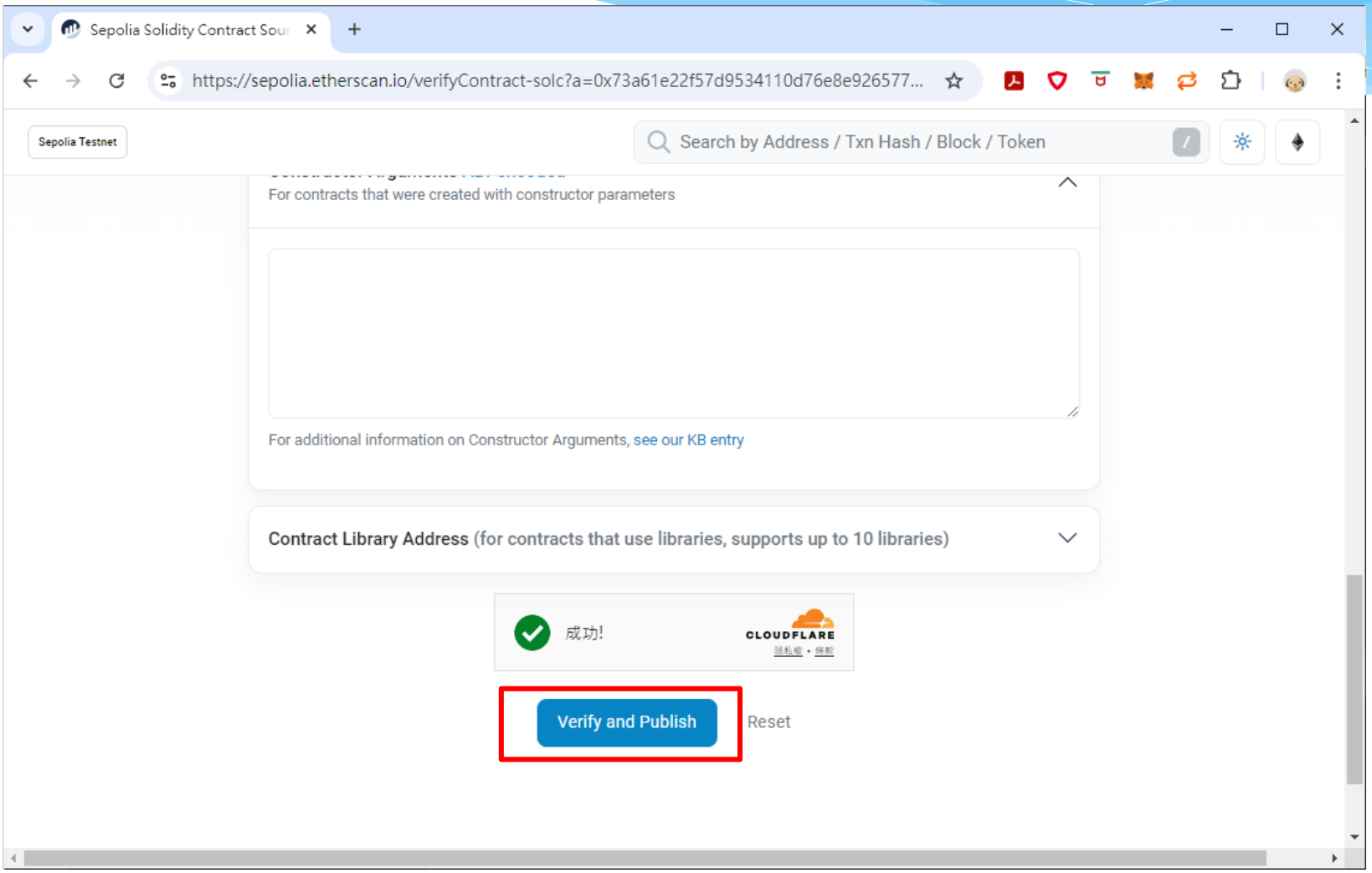
View & Publish Contract Source Code



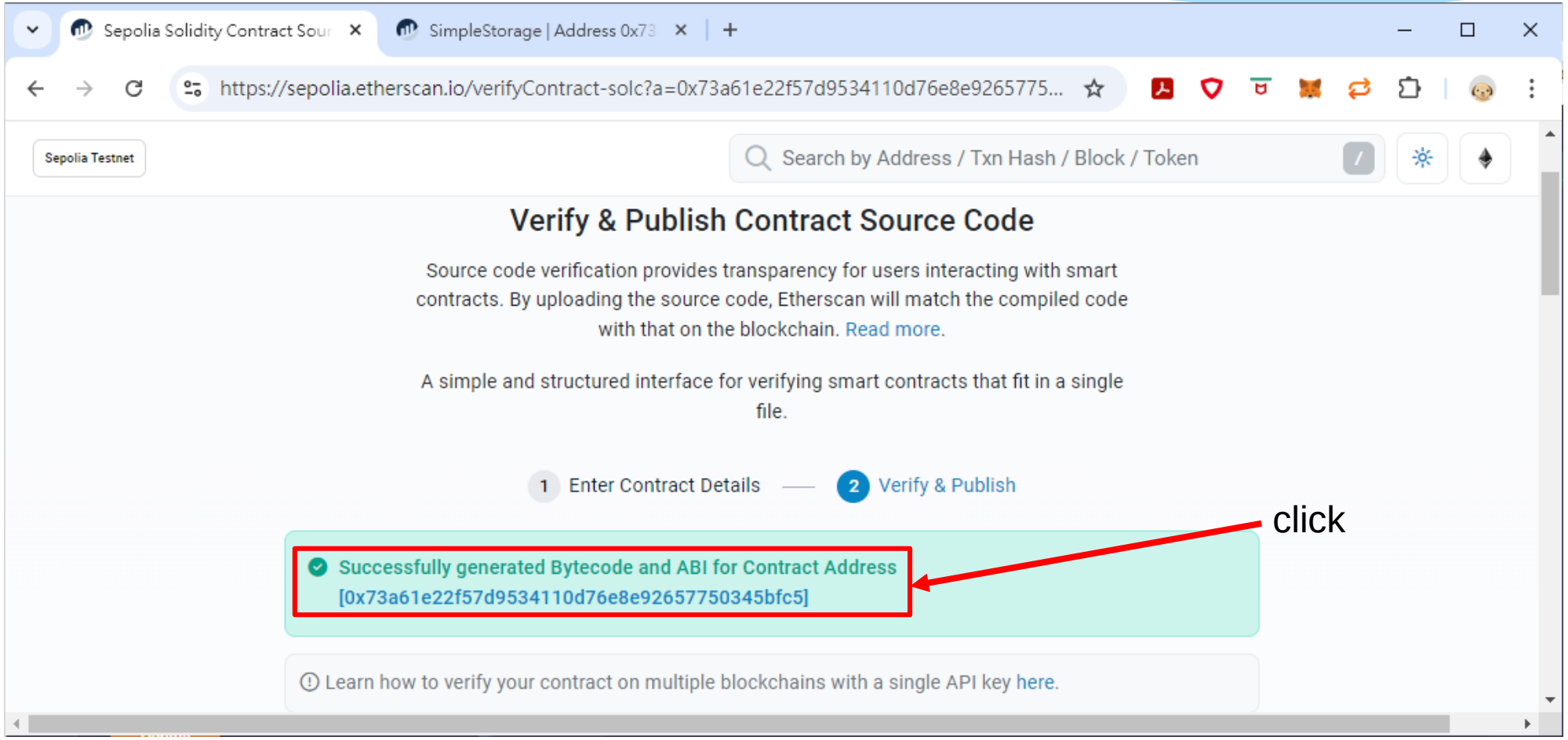
View & Publish Contract Source Code



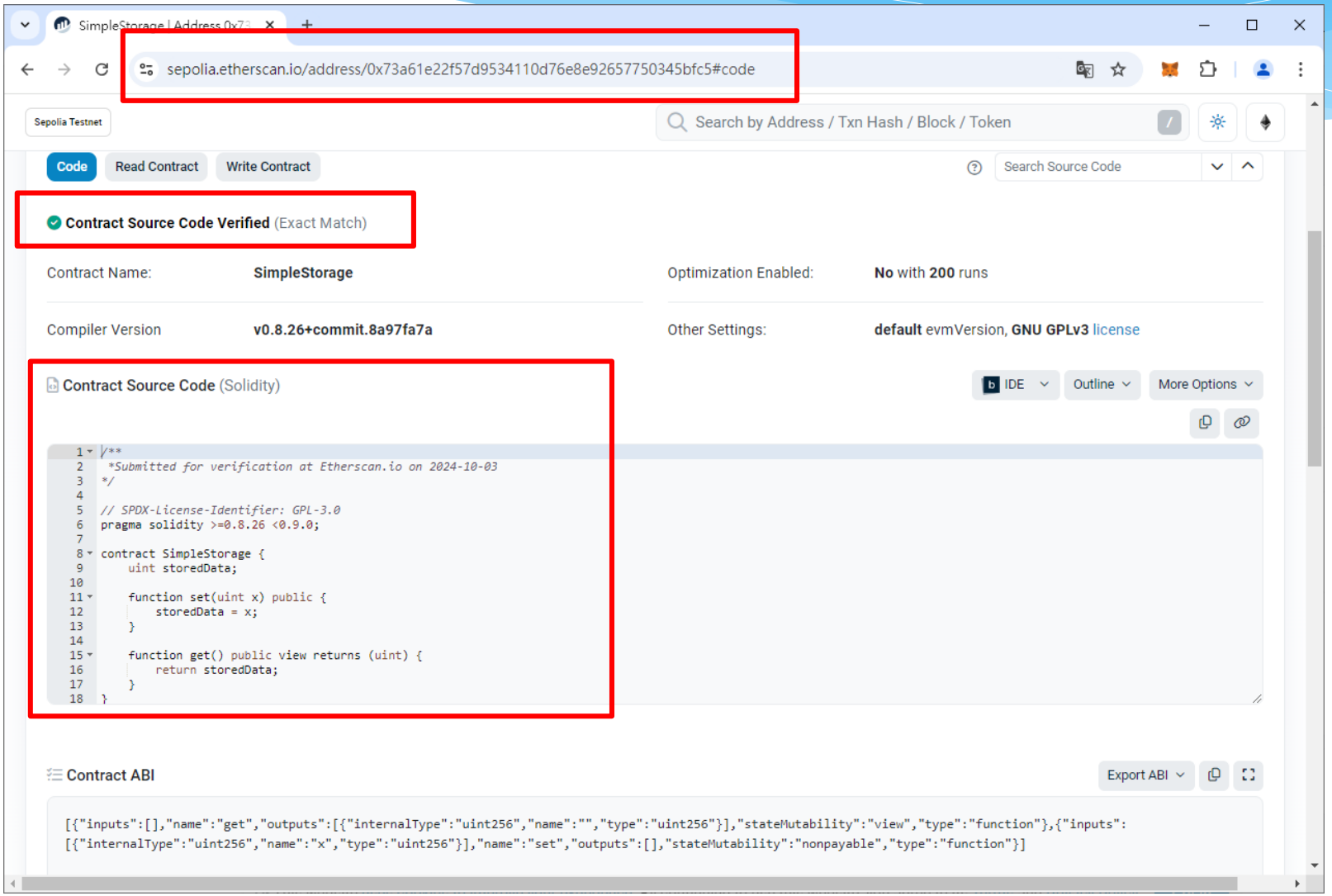
View & Publish Contract Source Code



View & Publish Contract Source Code

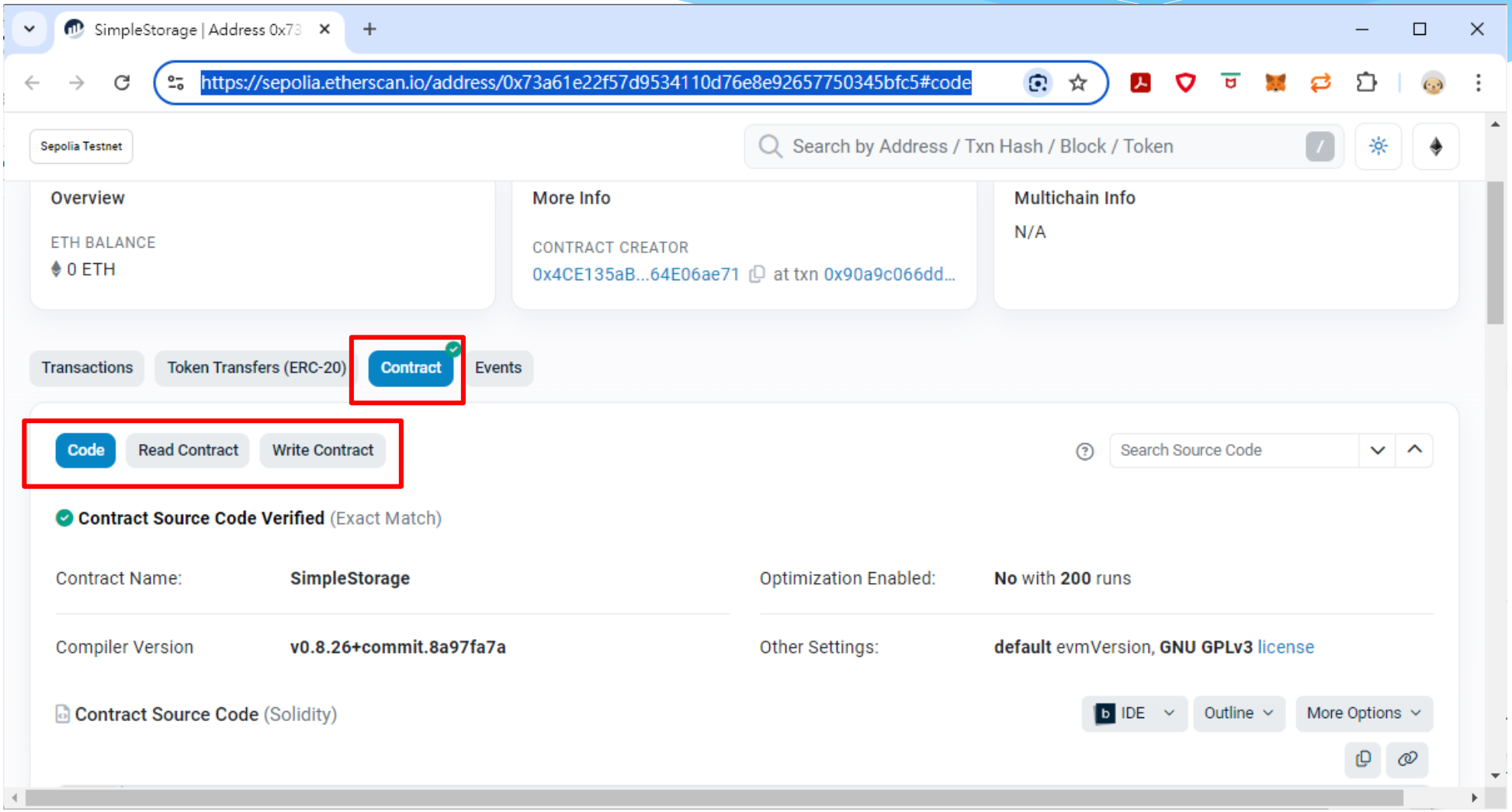


View Contract Source Code



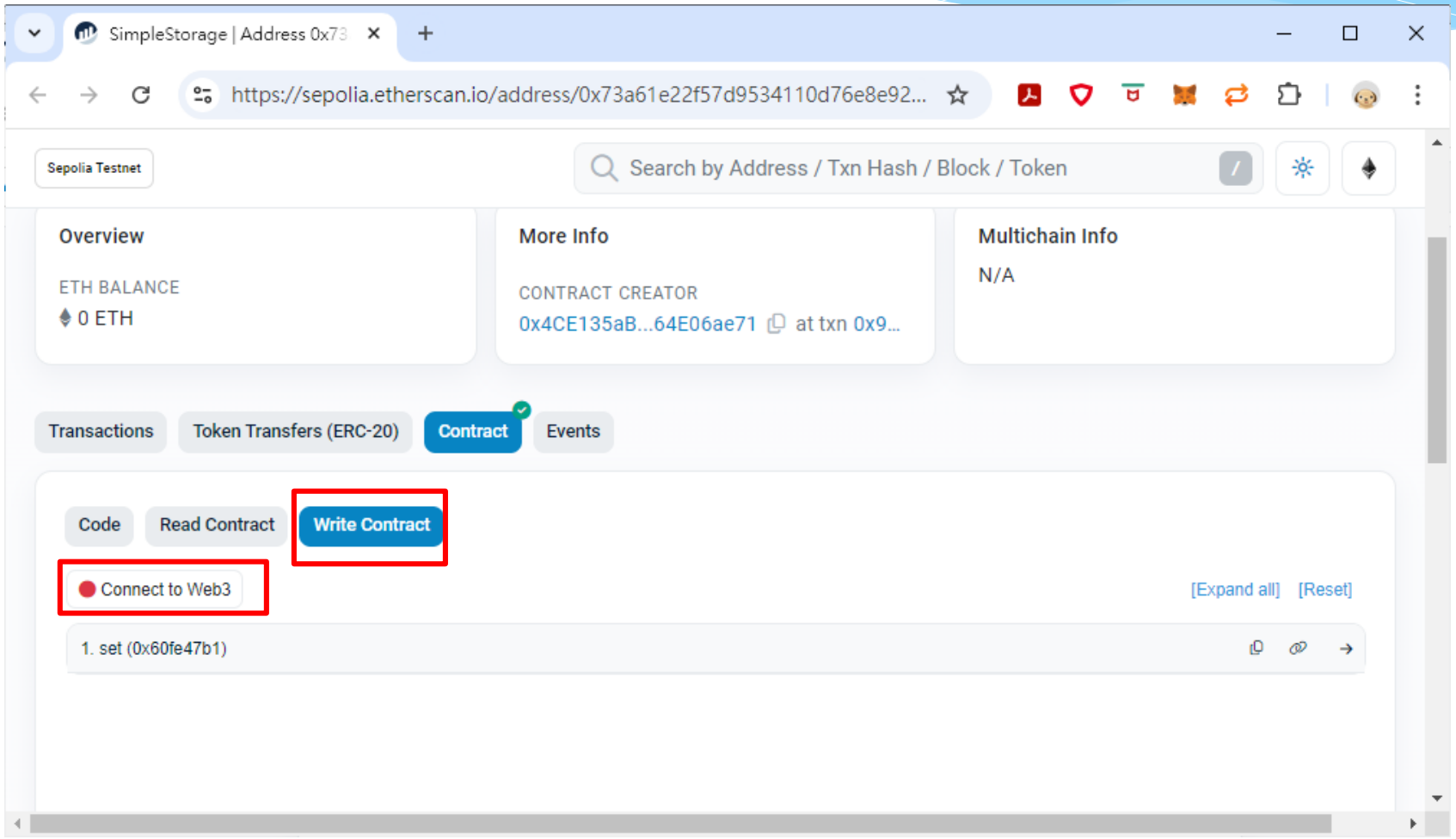
https://sepolia.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5#code

View & Publish Contract Source Code



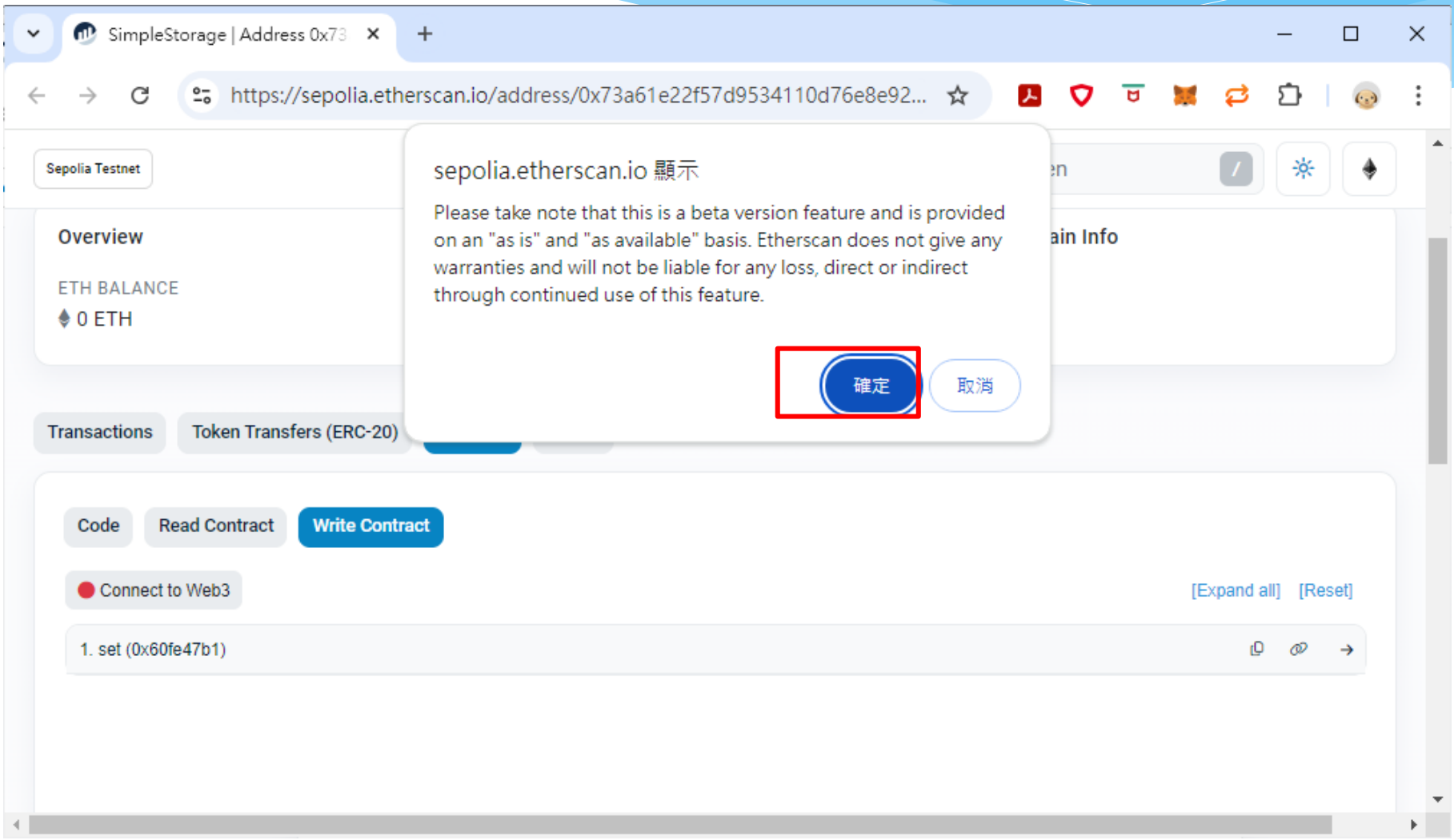
https://sepolia.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5#code

Write Contract-Connected Web3



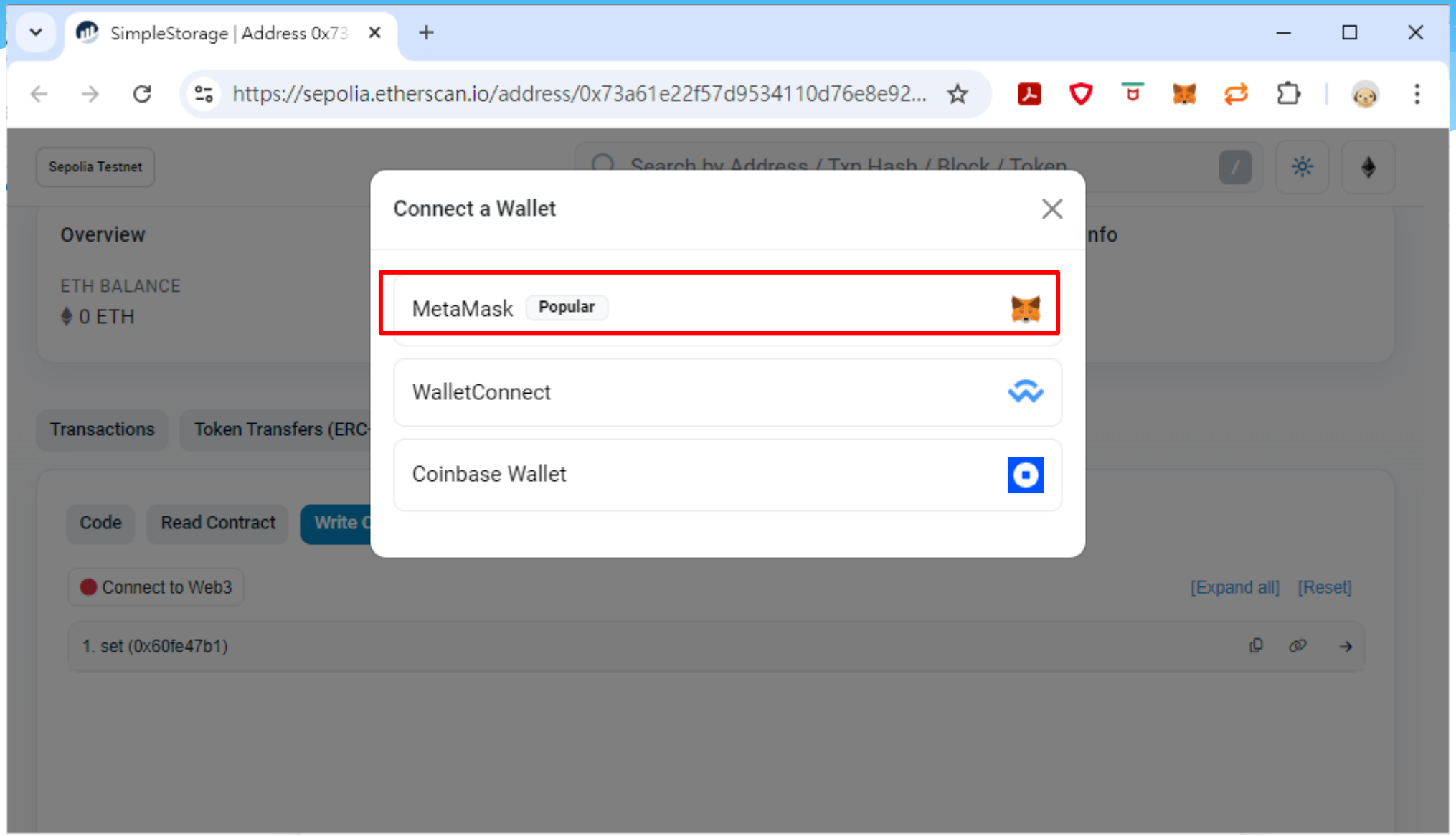
https://sepolia.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5#code

Connected-Web3



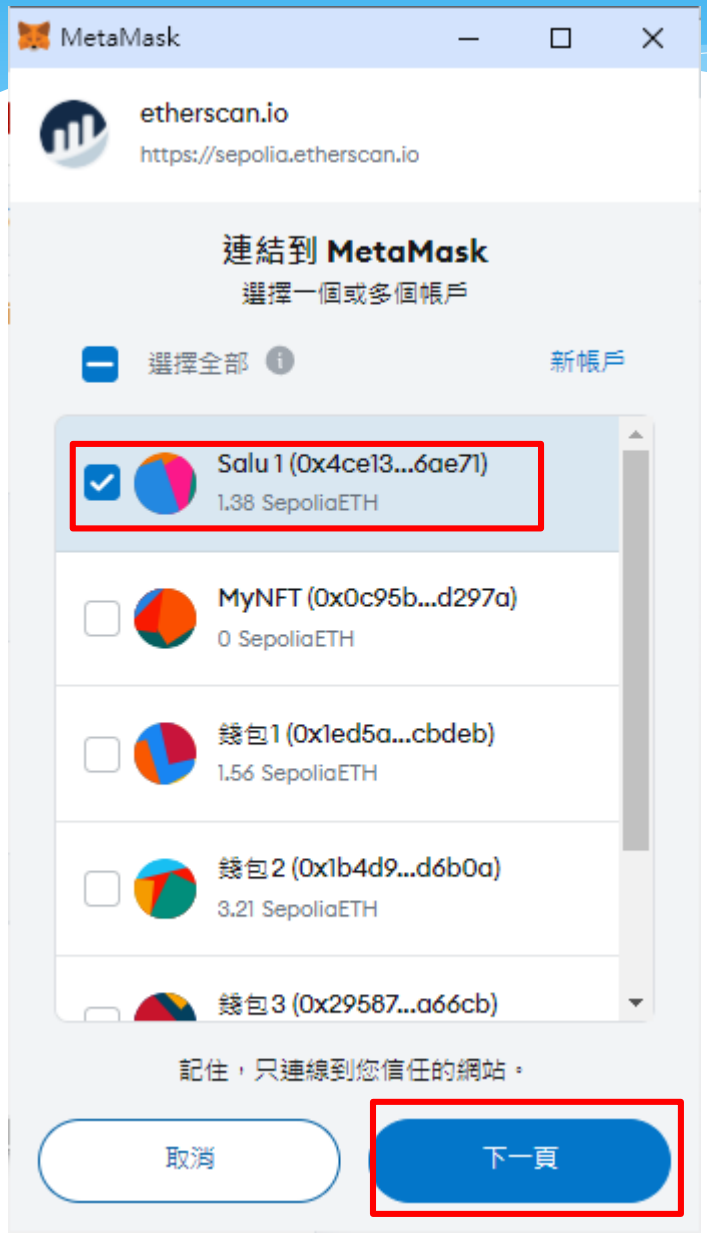
https://sepolia.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5#code

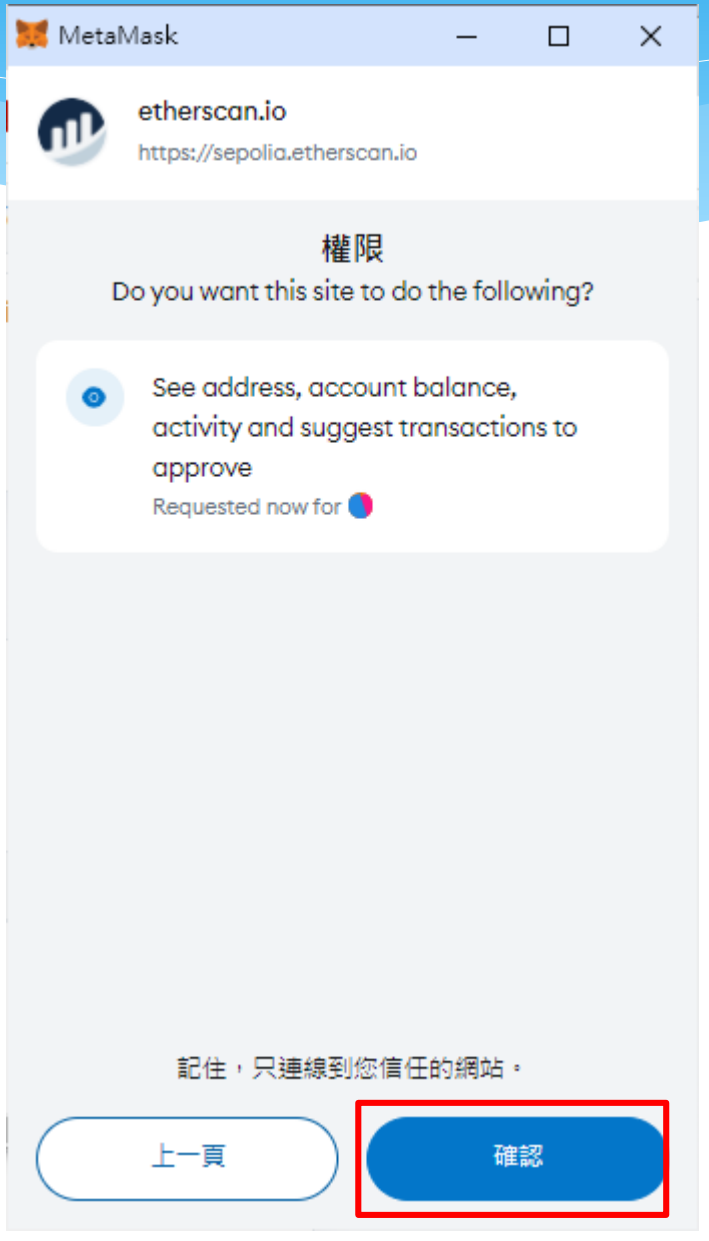
Connected-Web3



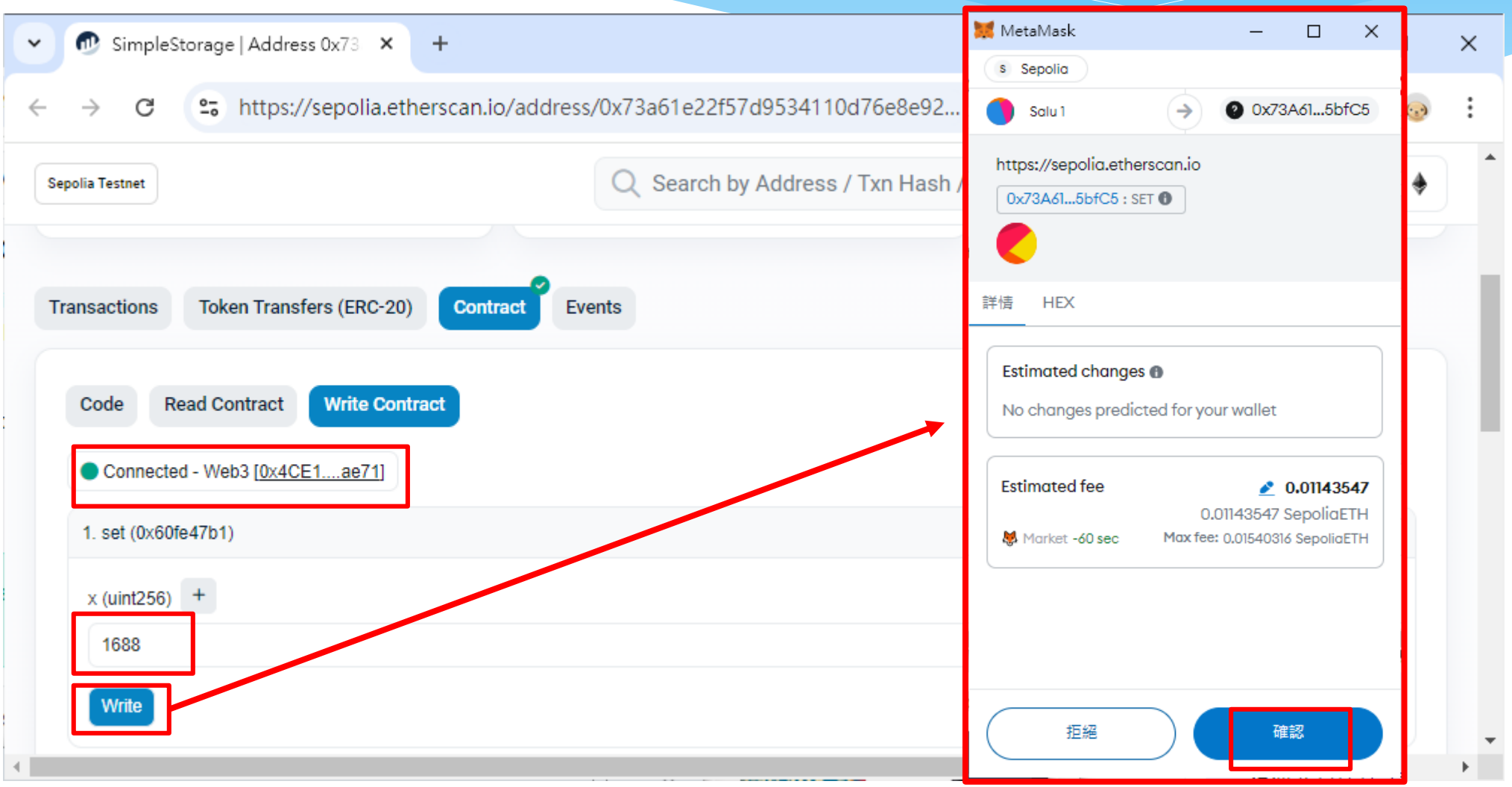
https://sepolia.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5#code

Connected-Web3

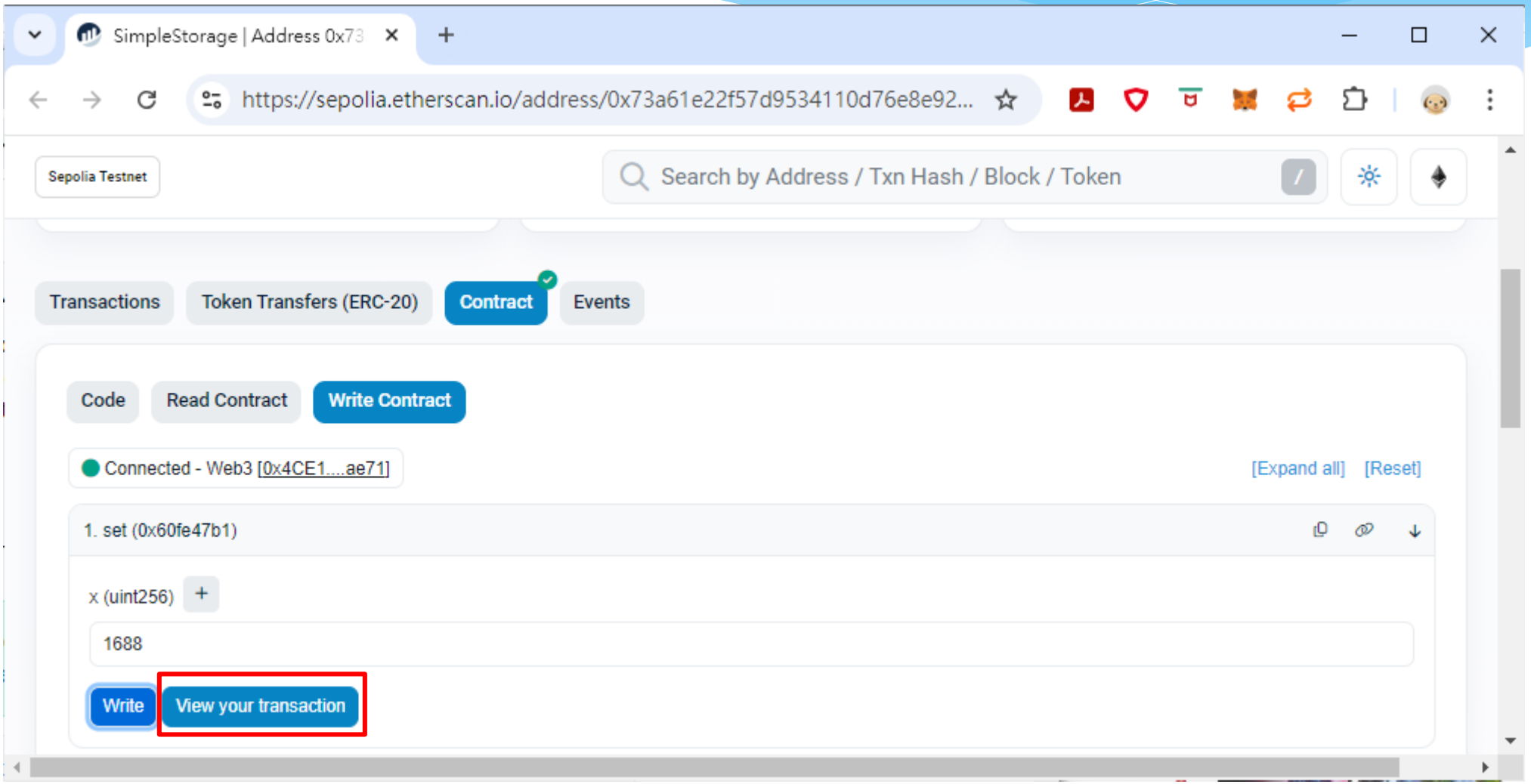




Write contract-set function

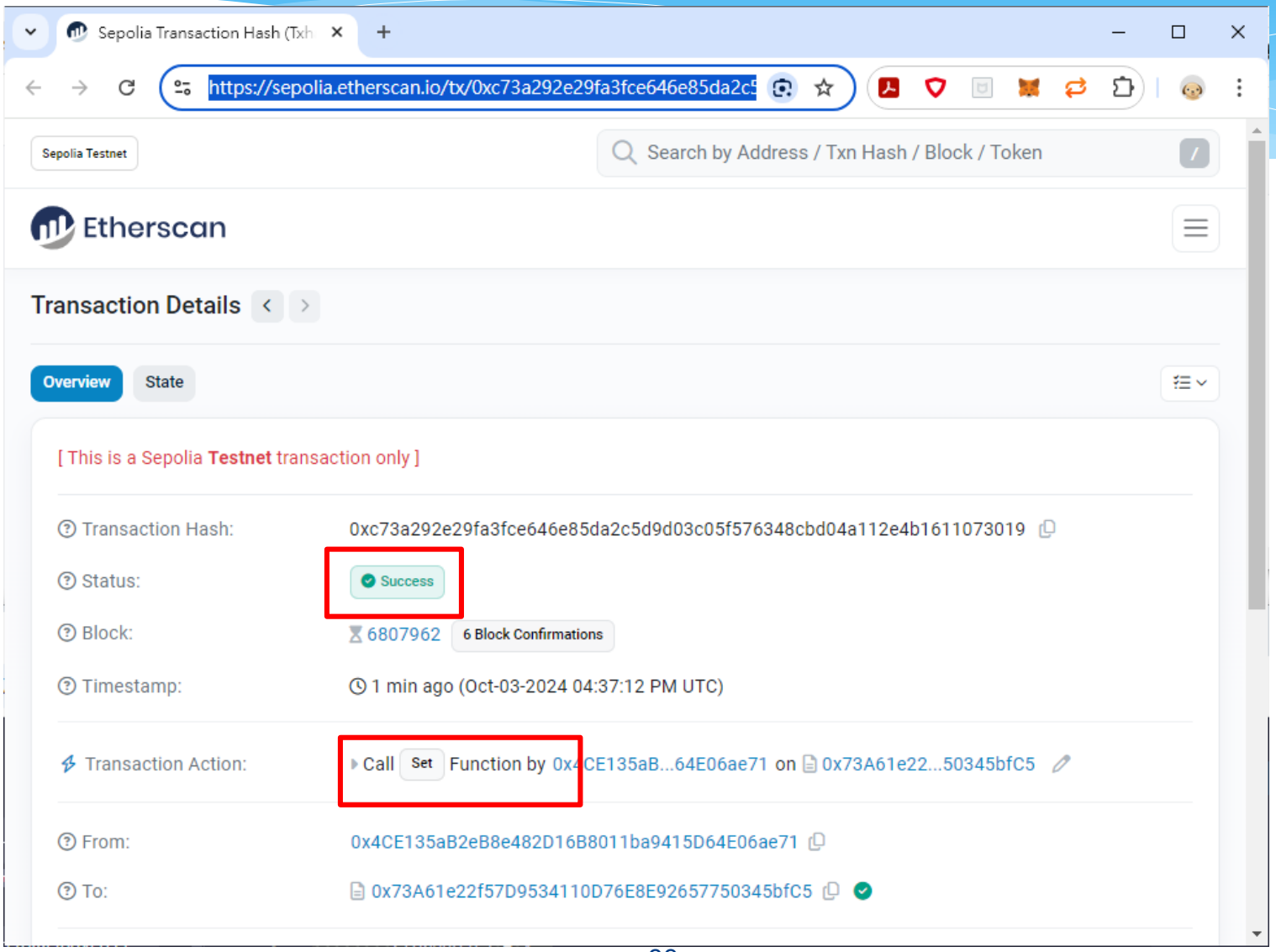


View your transaction



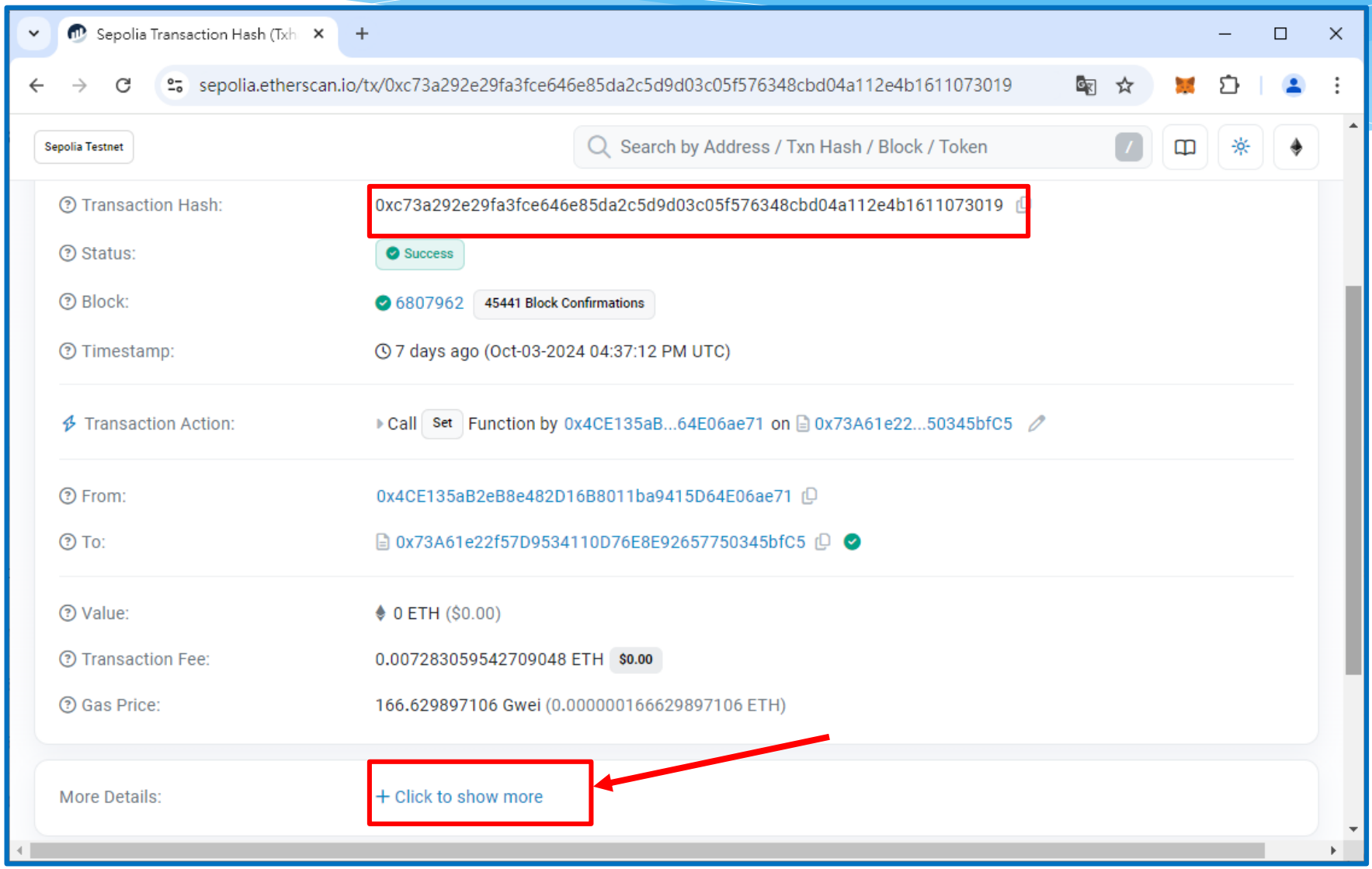
https://sepolia.etherscan.io/tx/0xc73a292e29fa3fce646e85da2c5d9d03c05f576348cbd04a112e4b1611073019

View your transaction



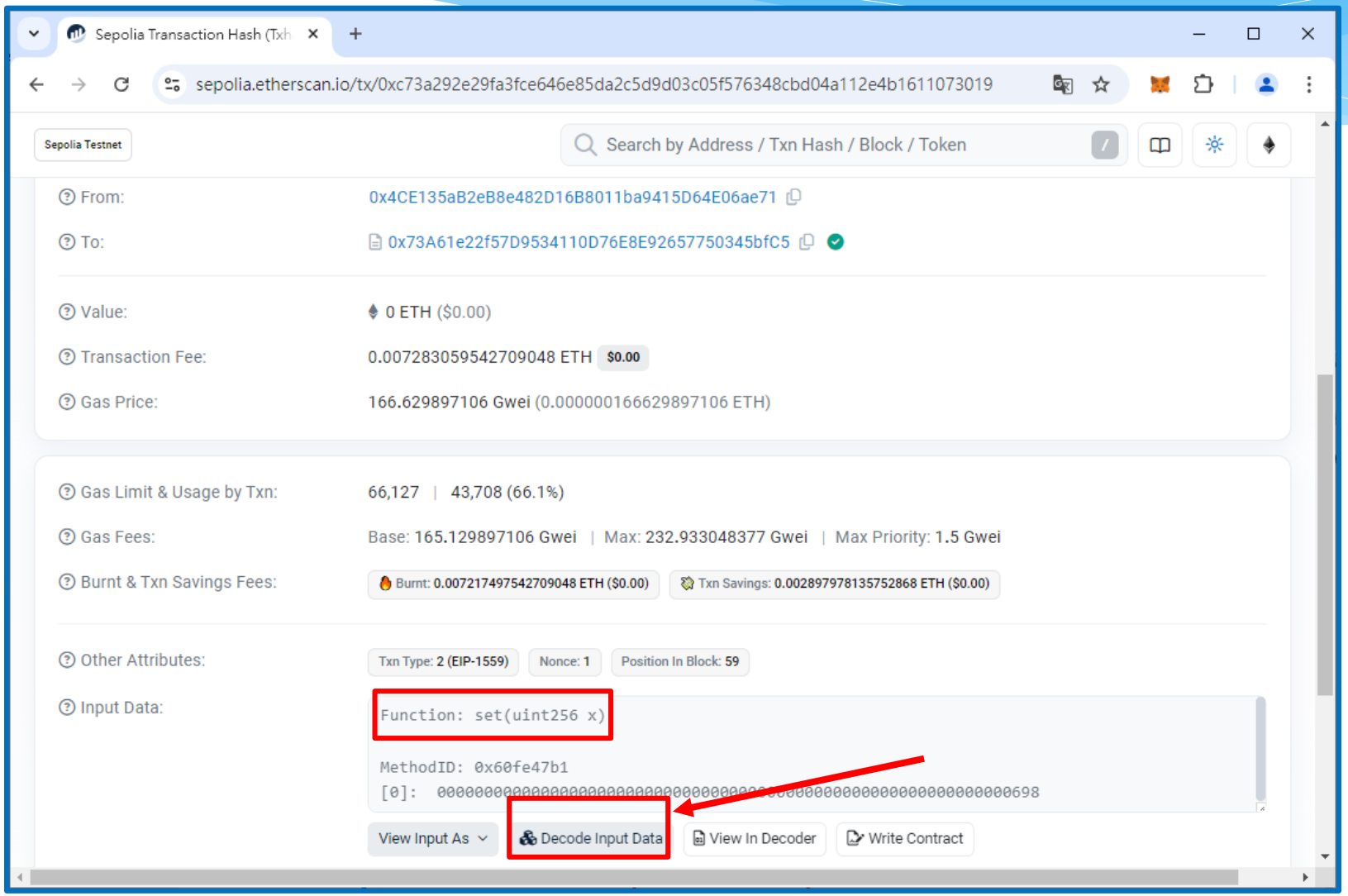
https://sepolia.etherscan.io/tx/0xc73a292e29fa3fce646e85da2c5d9d03c05f576348cbd04a112e4b1611073019

View your transaction-More Detail



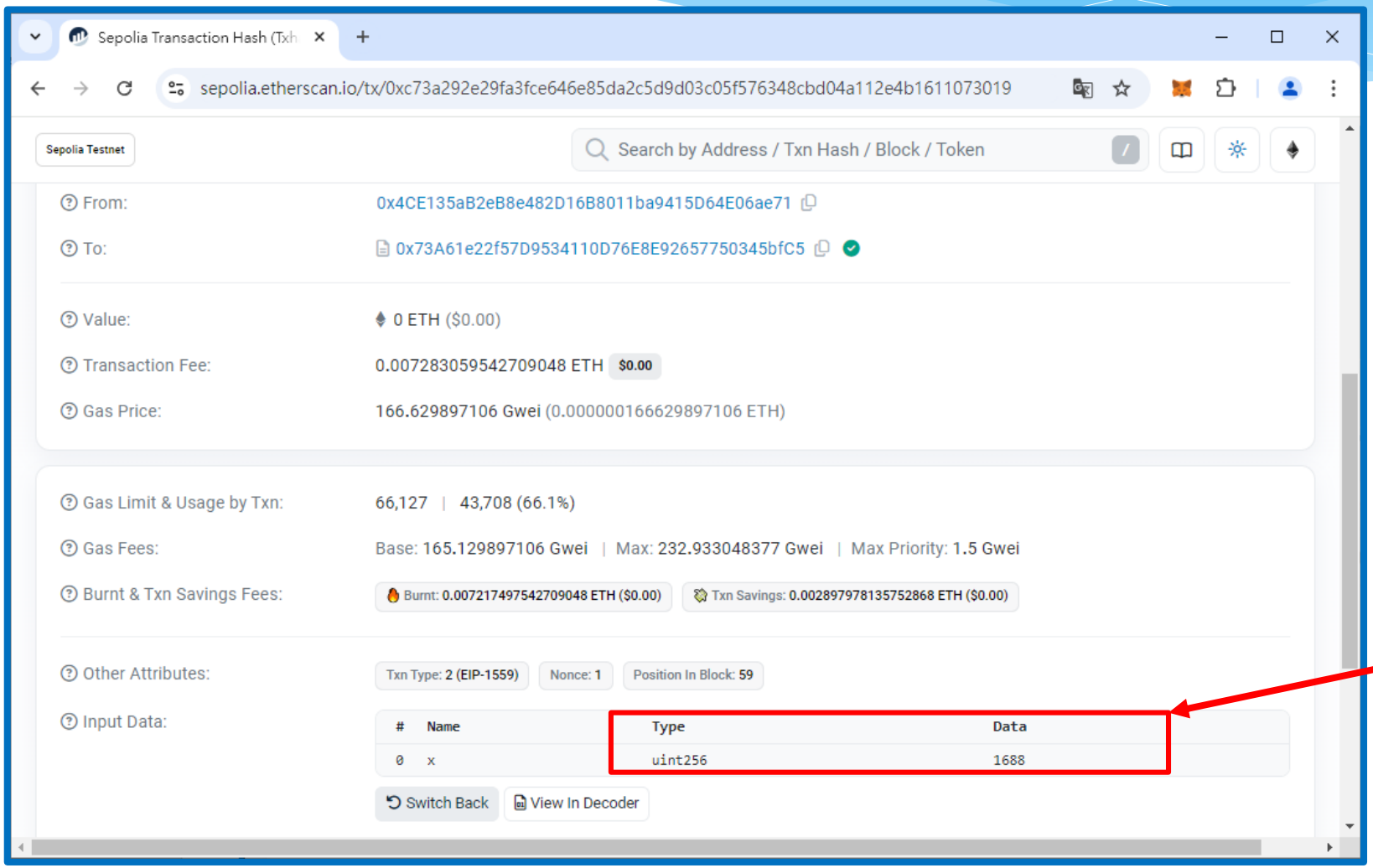
https://sepolia.etherscan.io/tx/0xc73a292e29fa3fce646e85da2c5d9d03c05f576348cbd04a112e4b1611073019

View your transaction-Input Data



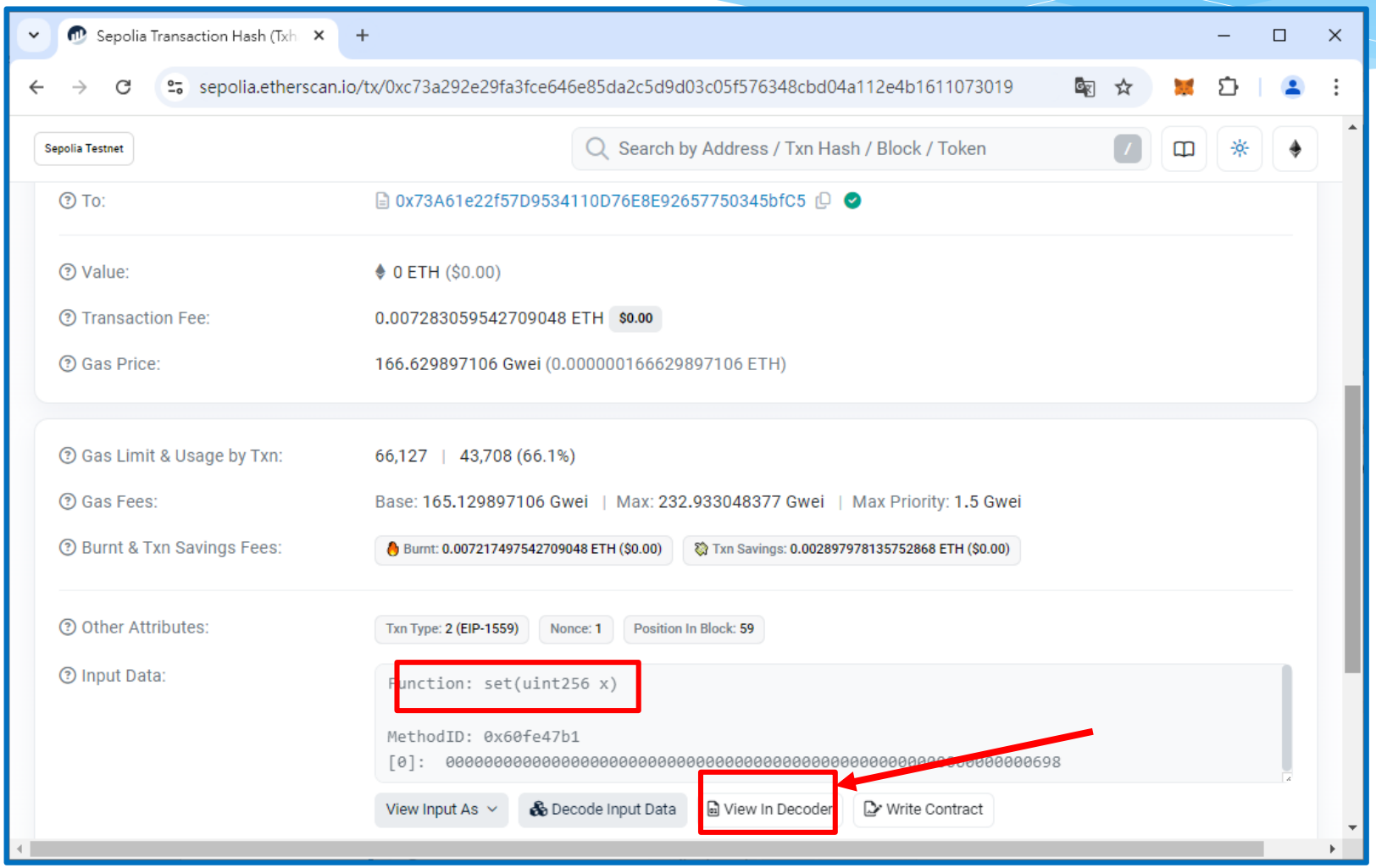
https://sepolia.etherscan.io/tx/0xc73a292e29fa3fce646e85da2c5d9d03c05f576348cbd04a112e4b1611073019

View your transaction-Decode Input Data



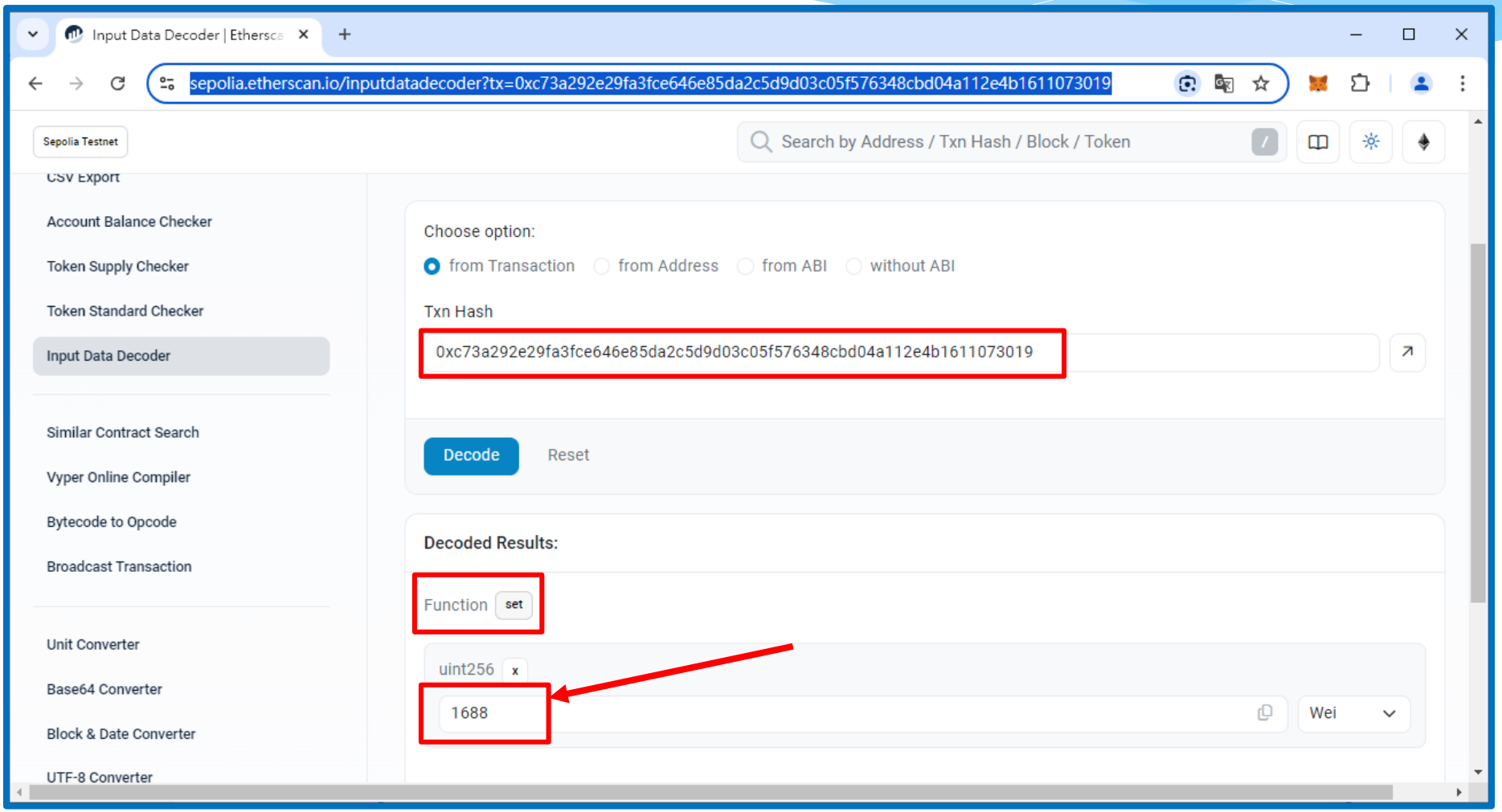
https://sepolia.etherscan.io/tx/0xc73a292e29fa3fce646e85da2c5d9d03c05f576348cbd04a112e4b1611073019

View your transaction-Input Data



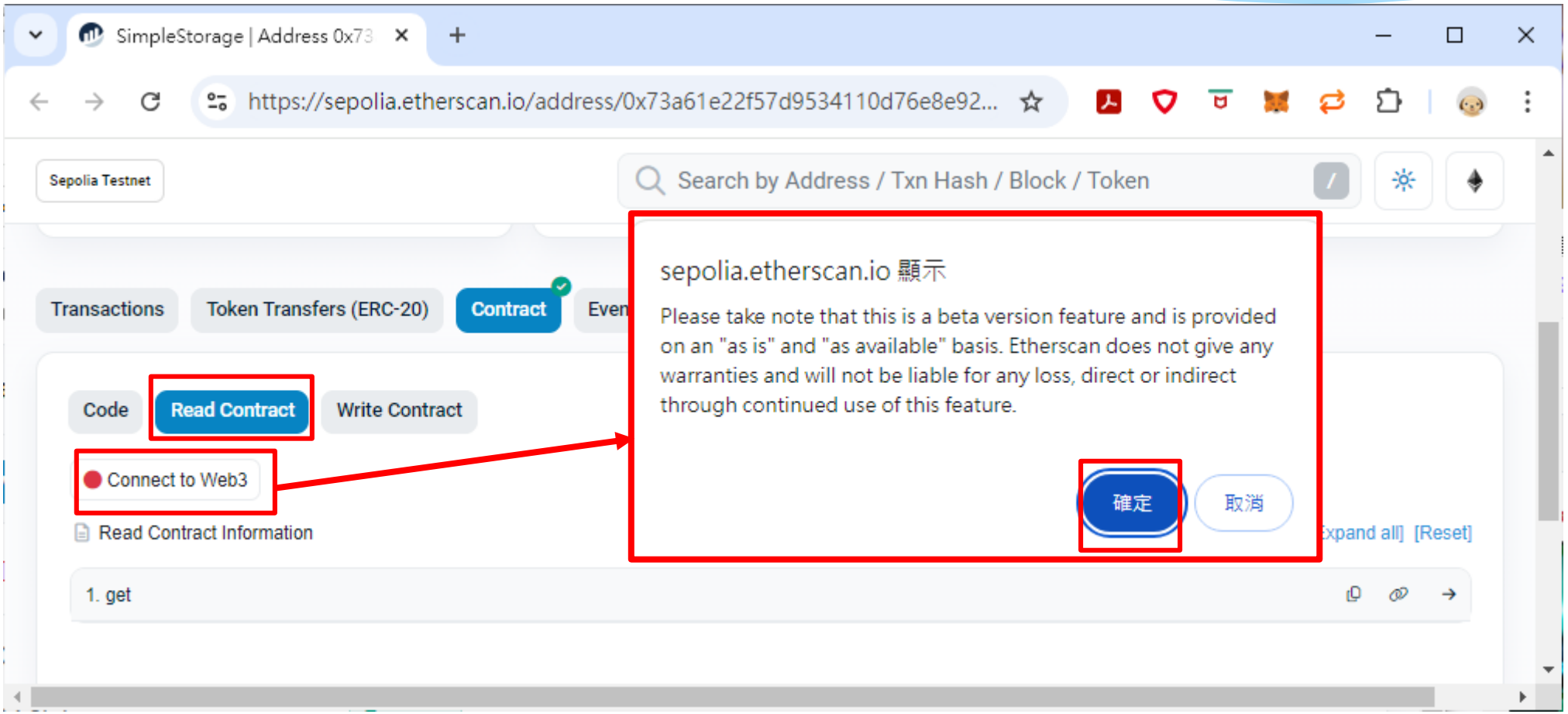
https://sepolia.etherscan.io/tx/0xc73a292e29fa3fce646e85da2c5d9d03c05f576348cbd04a112e4b1611073019

View your transaction-Input Data Decoder



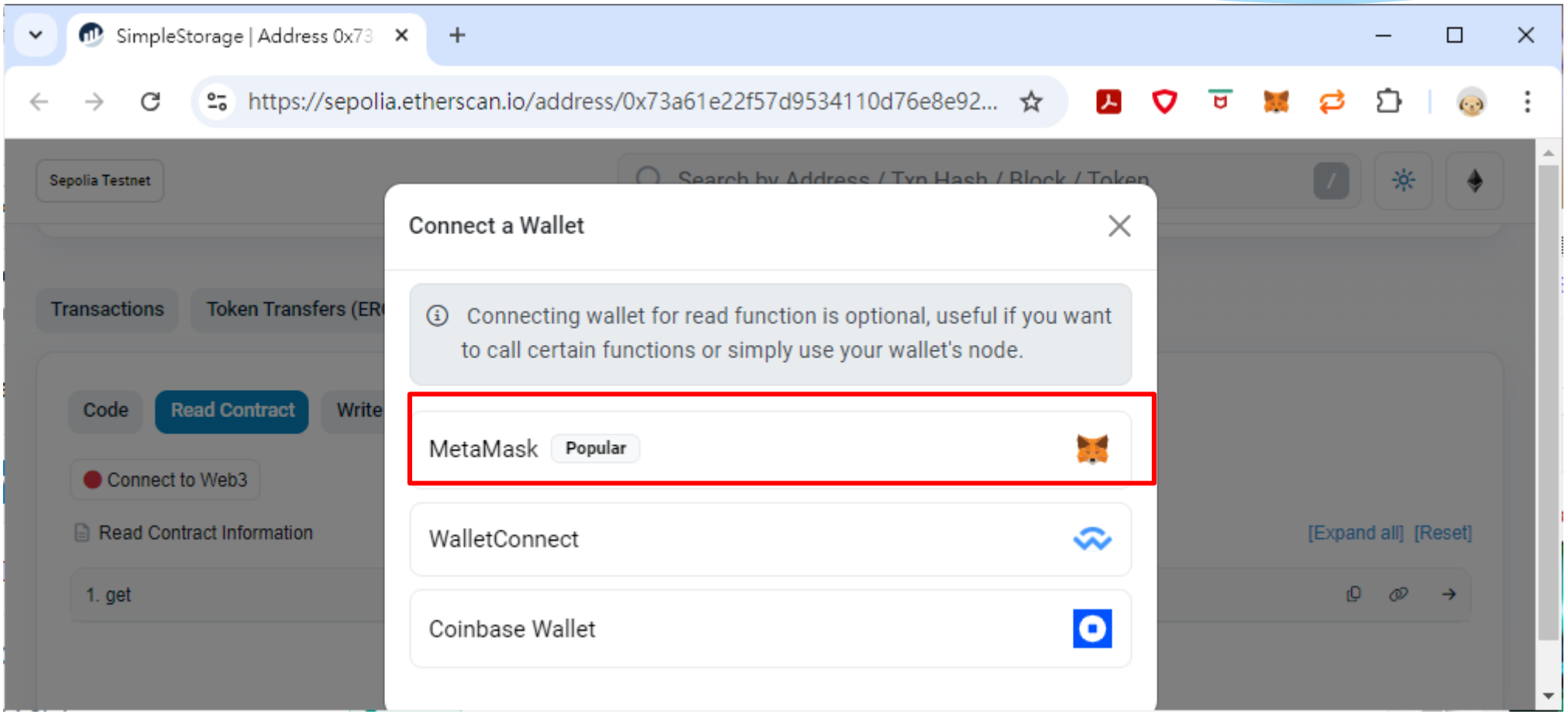
https://sepolia.etherscan.io/inputdatadecoder?tx=0xc73a292e29fa3fce646e85da2c5d9d03c05f576348cbd04a112e4b1611073019

Connect to Web3



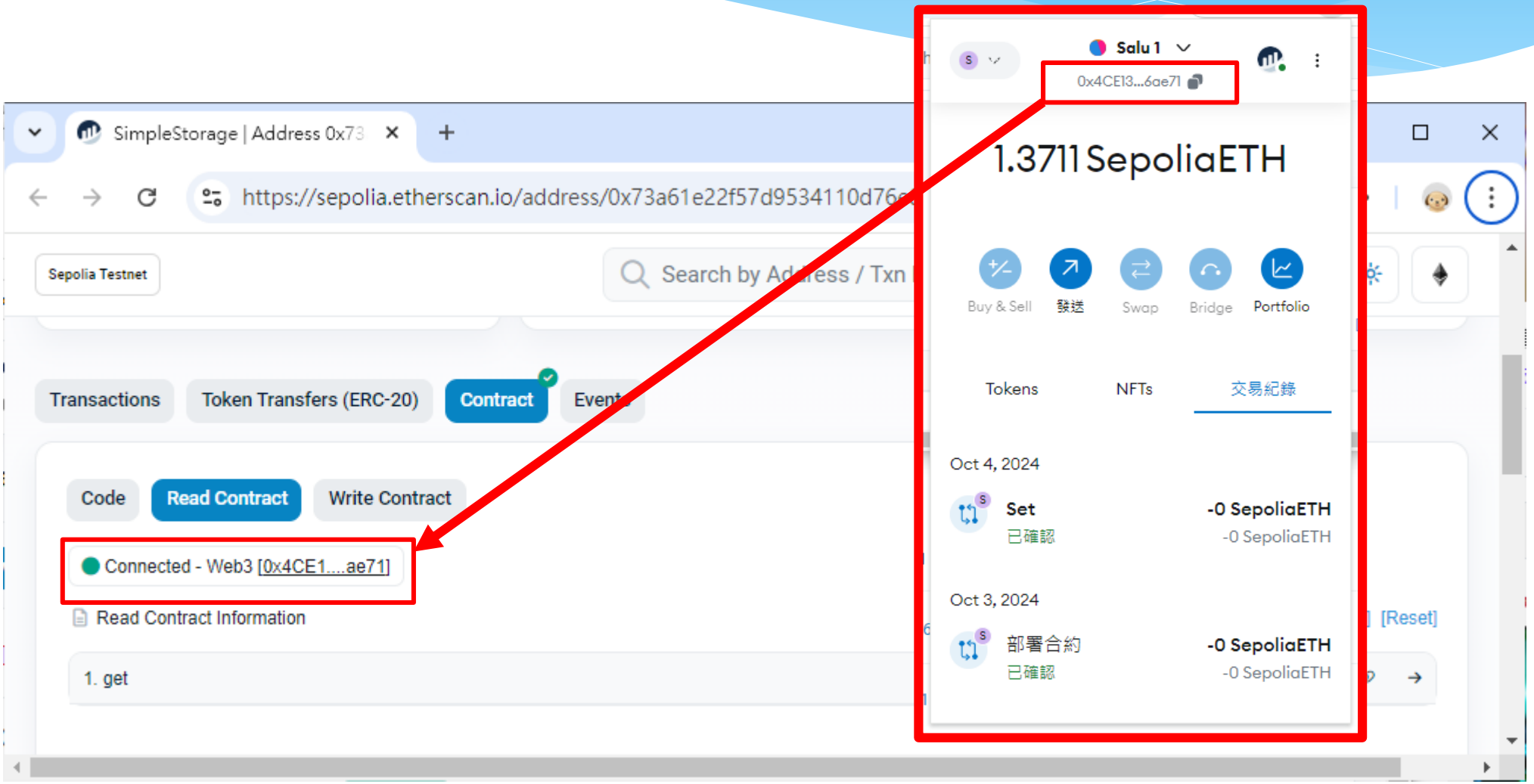
https://sepolia.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5#readContract

Connect a Wallet-MetaMask



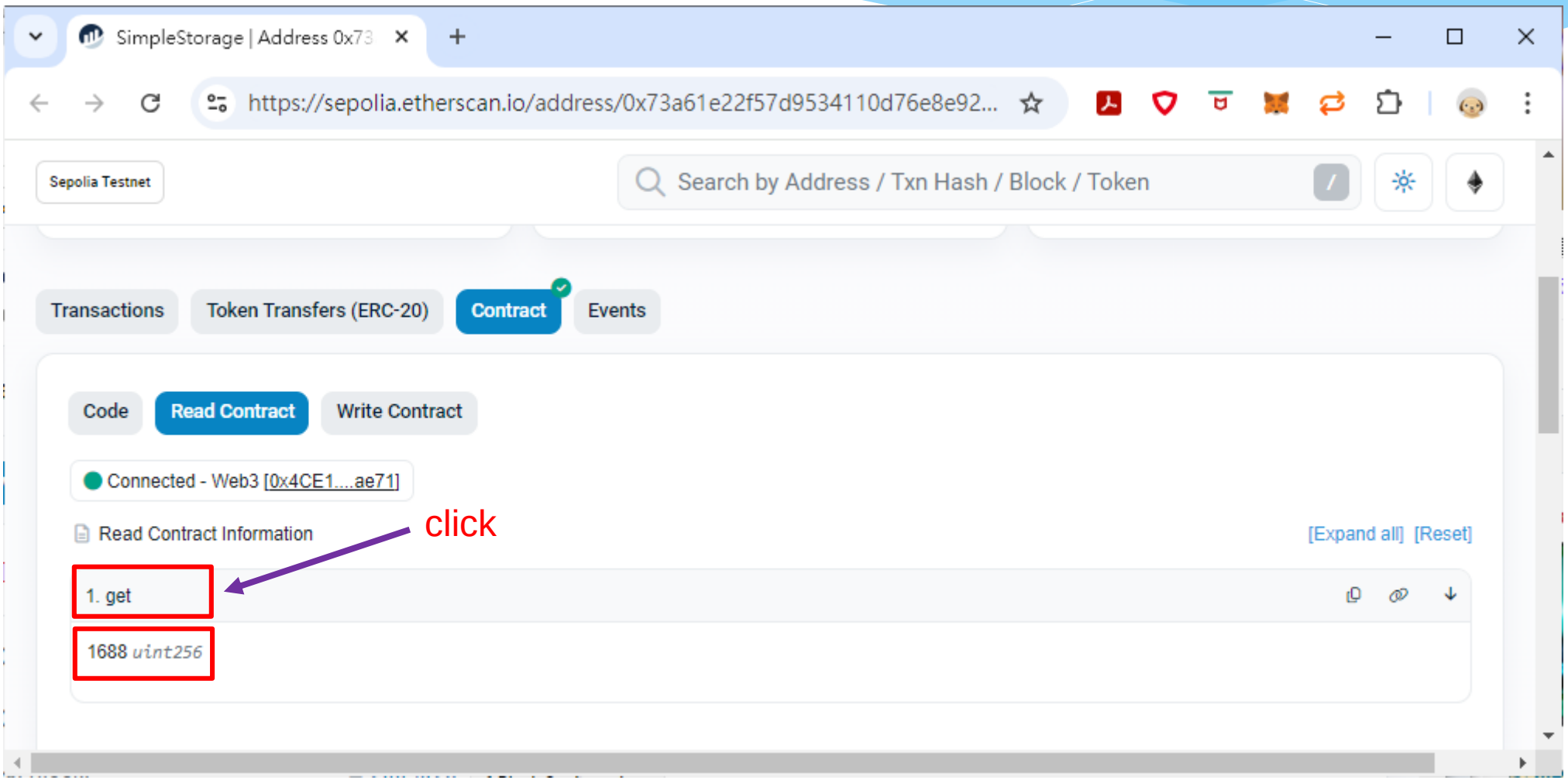
https://sepolia.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5#readContract

Connected Web3 using account



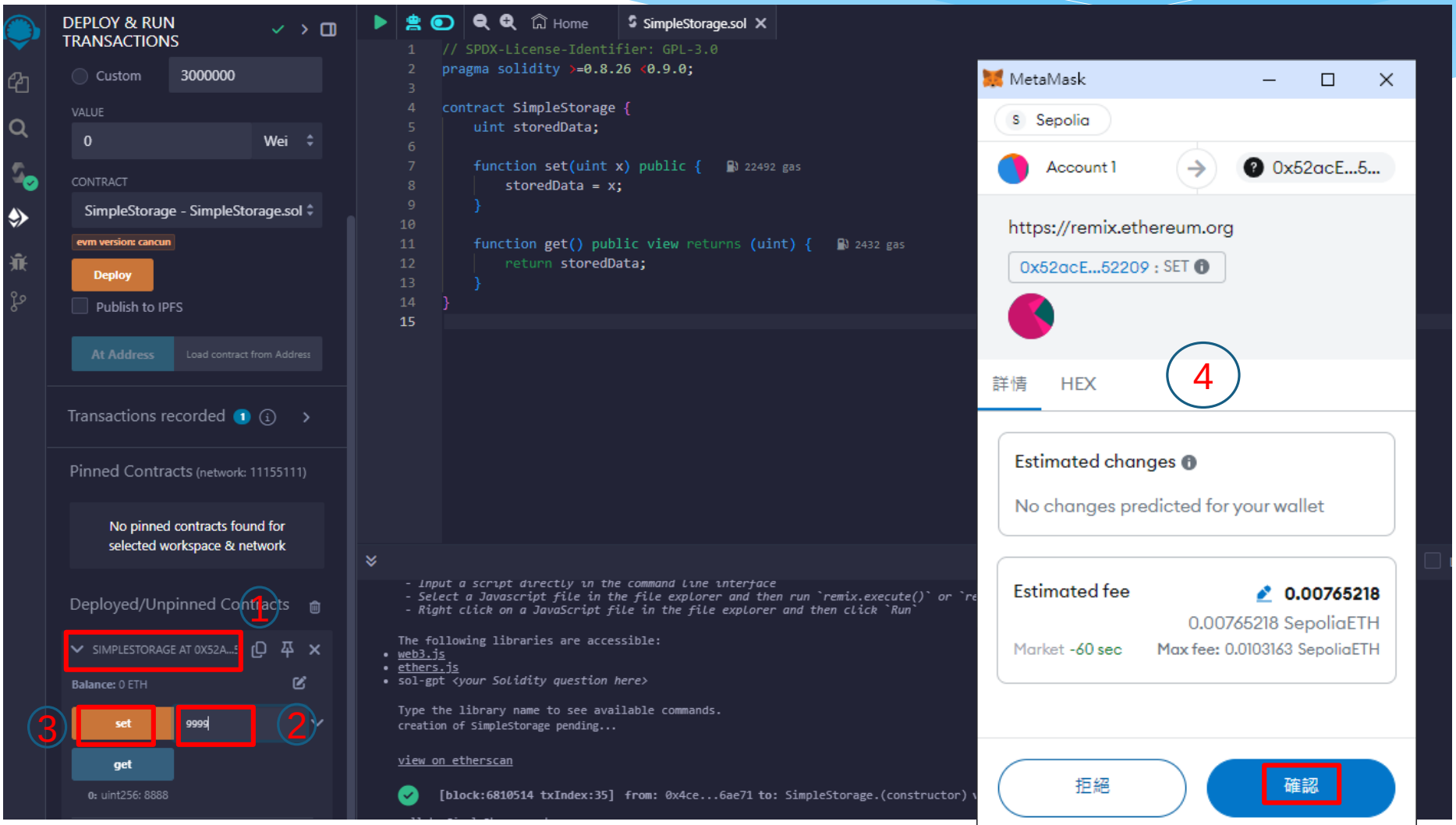
https://sepolia.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5#readContract

Read Contract-get function

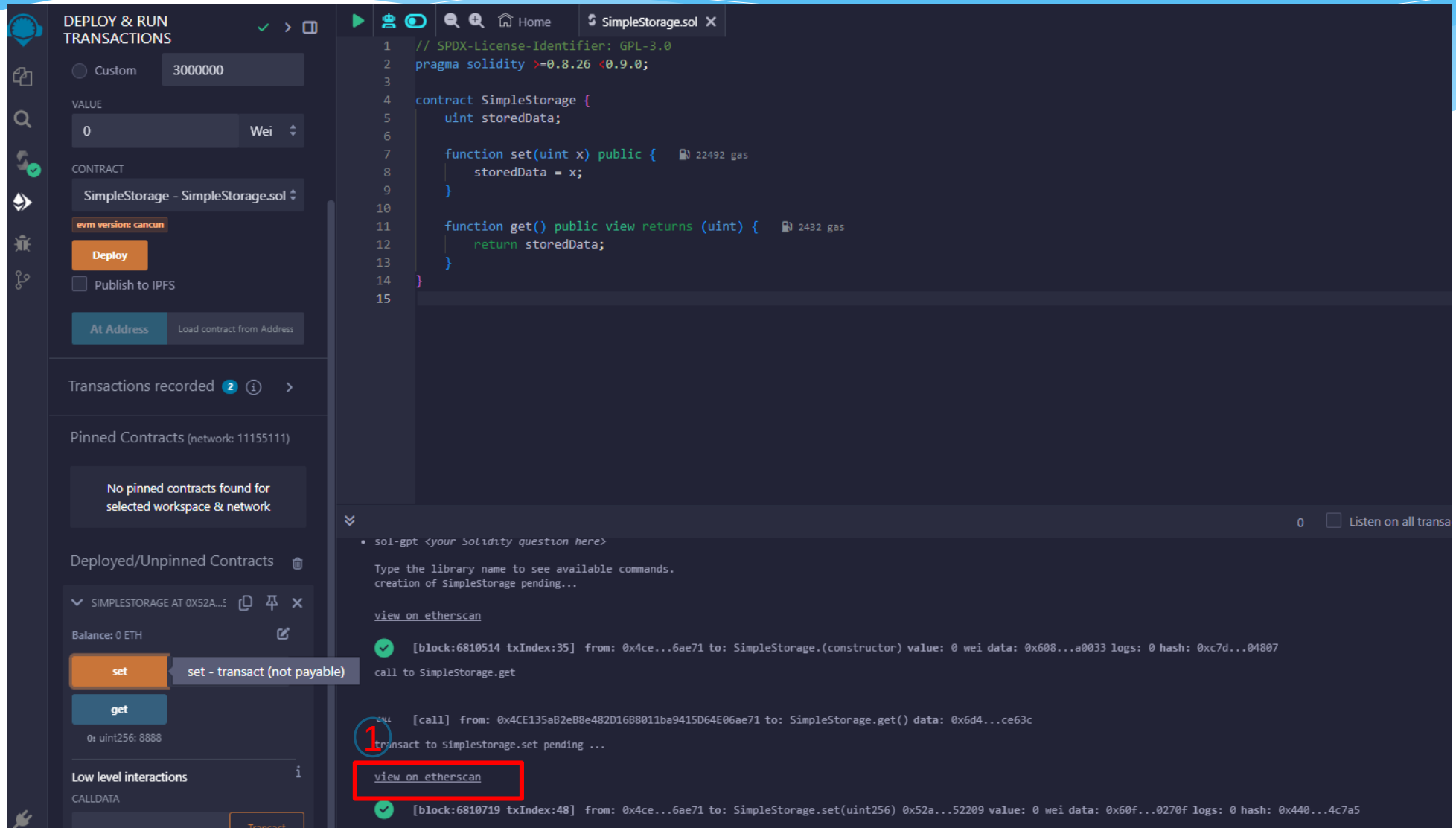


https://sepolia.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5#readContract

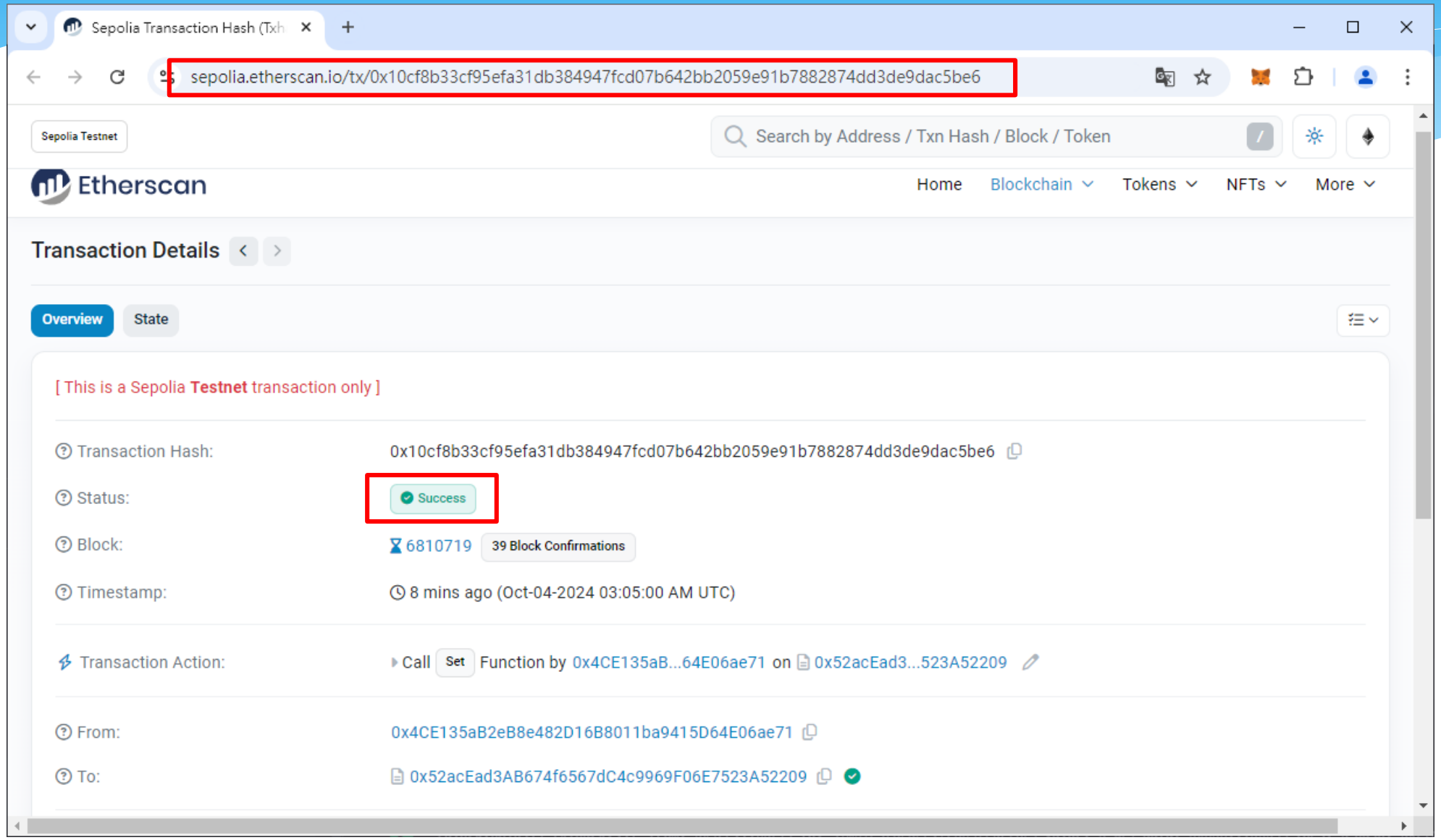
set Contract using Remix



View transaction on etherscan

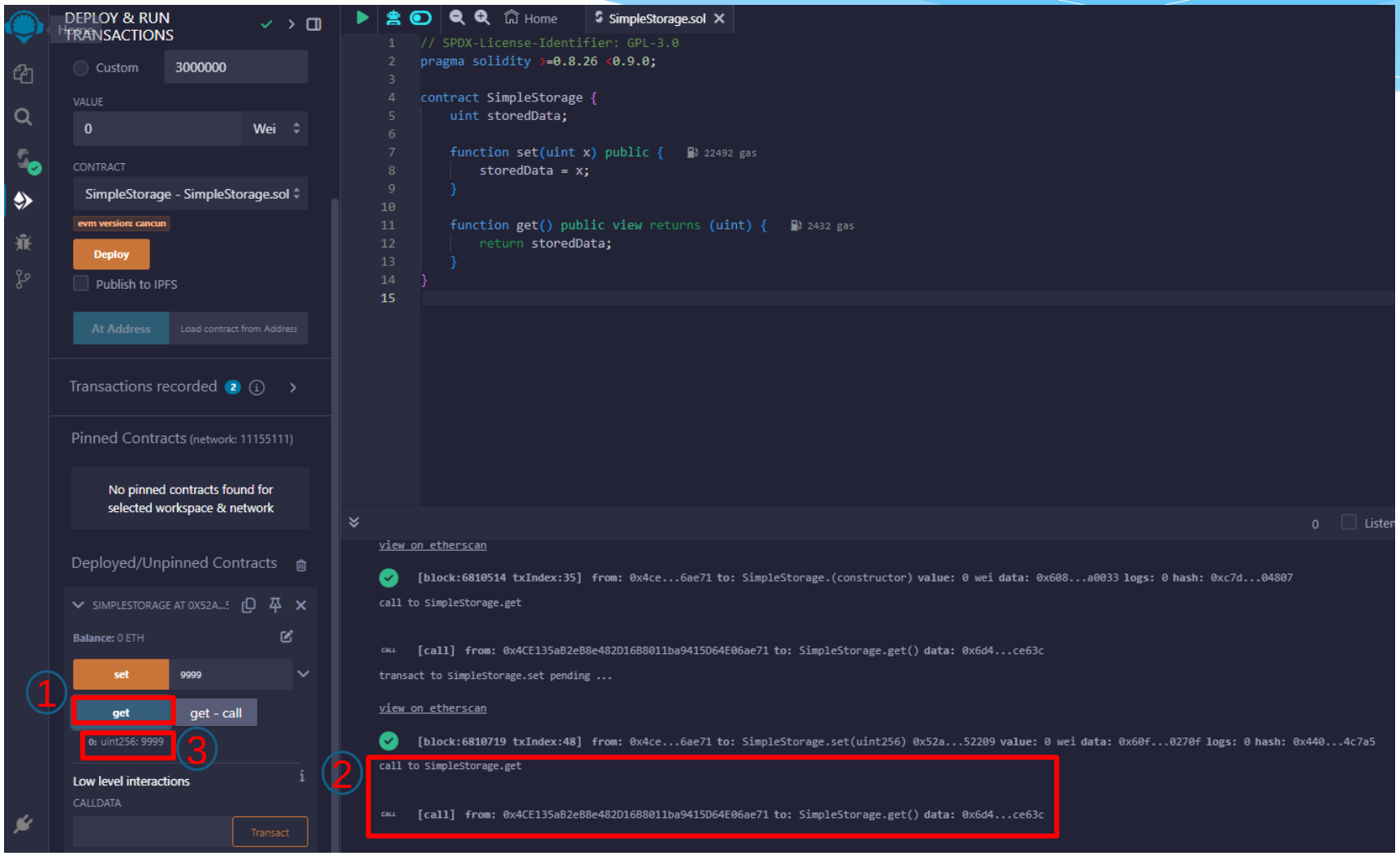


View transaction on etherscan



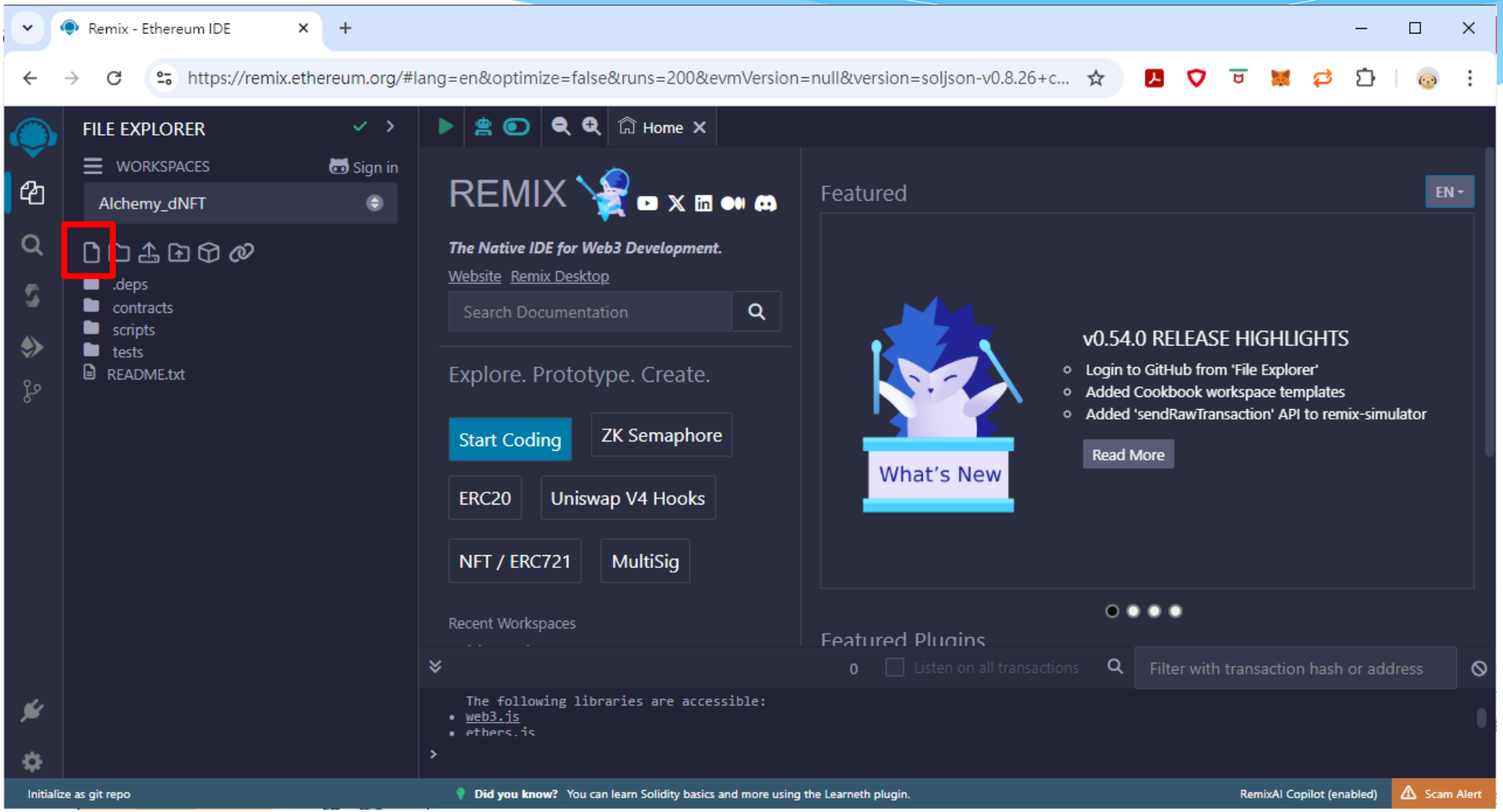
https://sepolia.etherscan.io/tx/0x10cf8b33cf95efa31db384947fcd07b642bb2059e91b7882874dd3de9dac5be6

get Contract using Remix



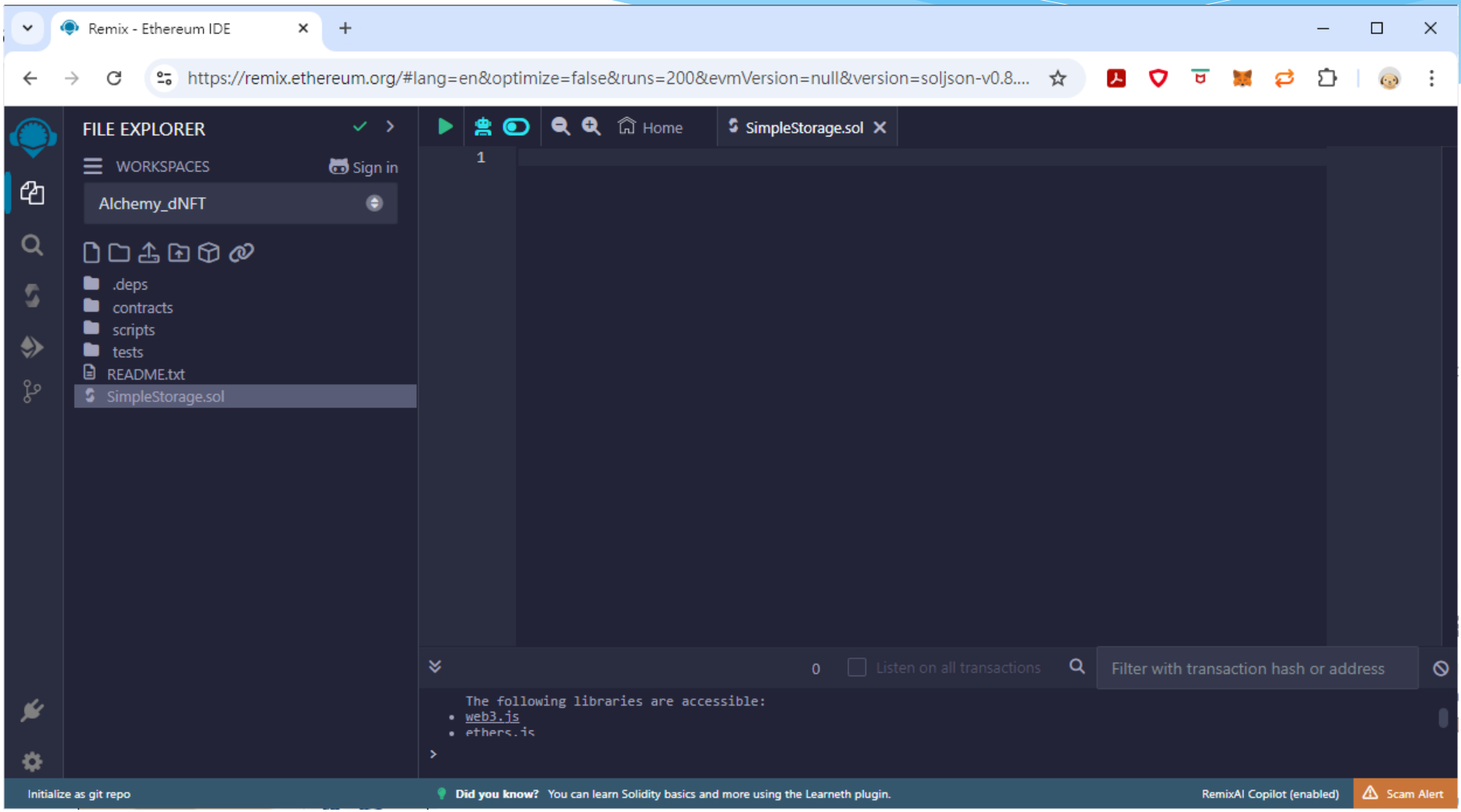
2. Deploy Smart Contract to Holesky Testnet

Create new file

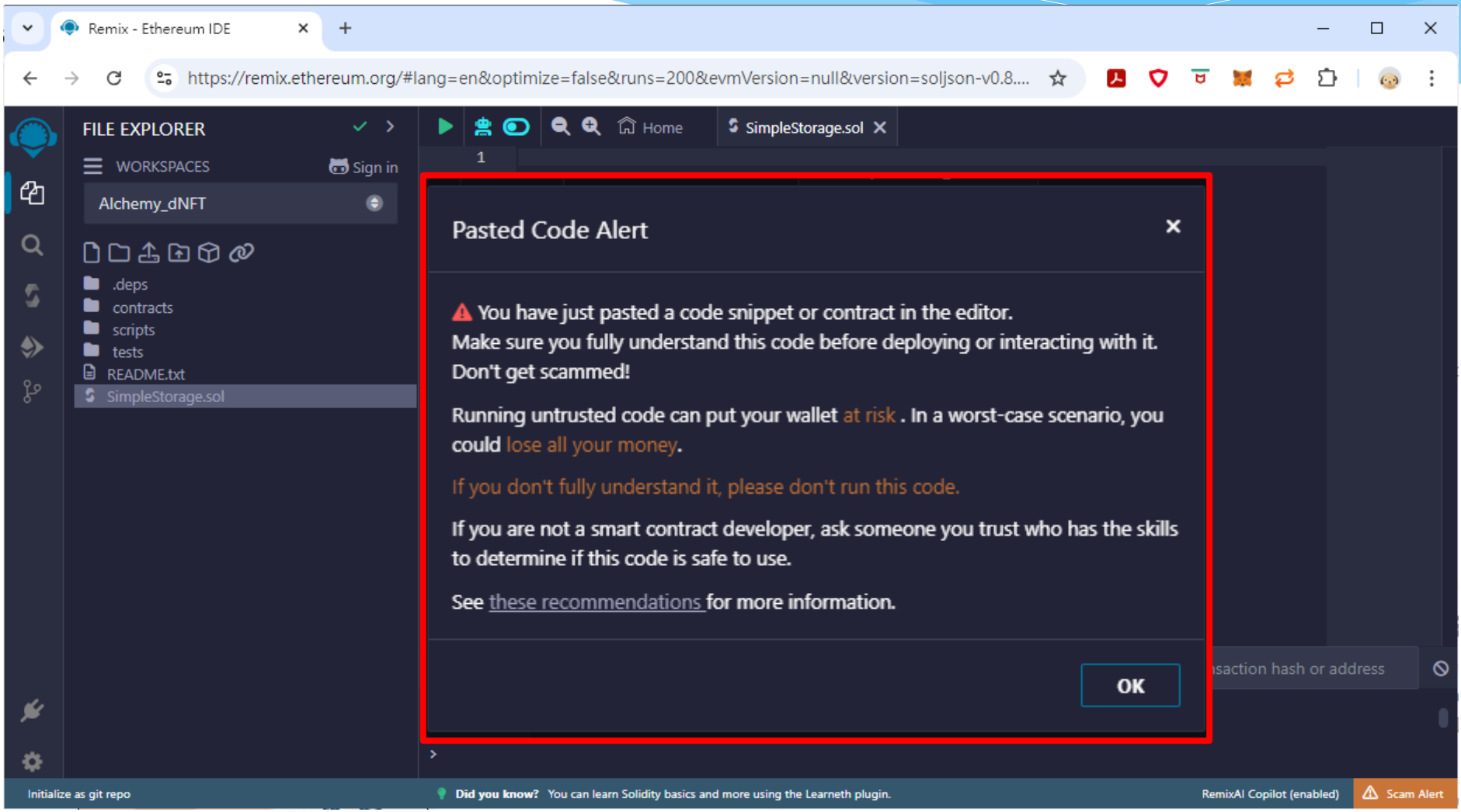


https://remix.ethereum.org

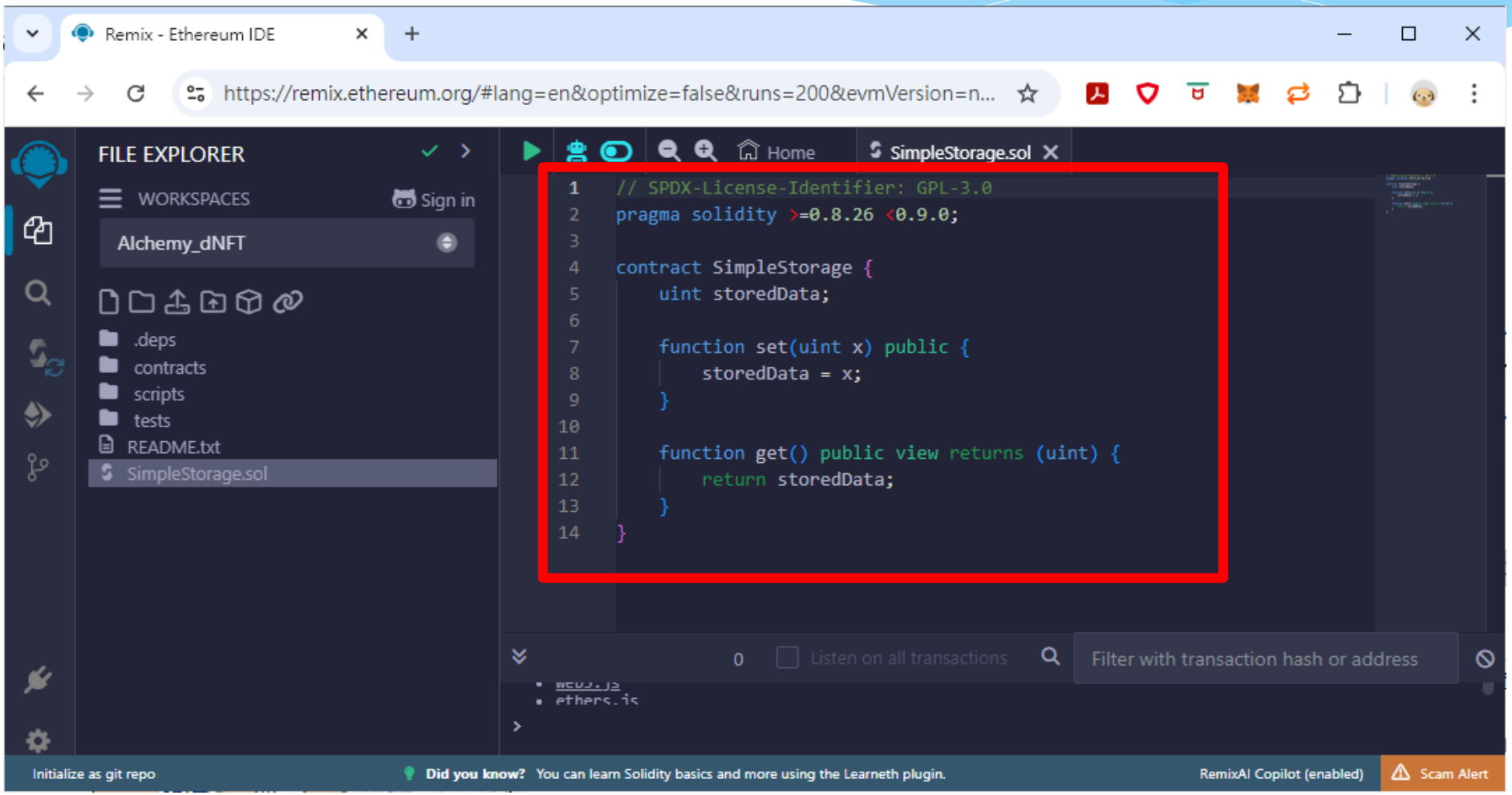
SimpleStorage.sol



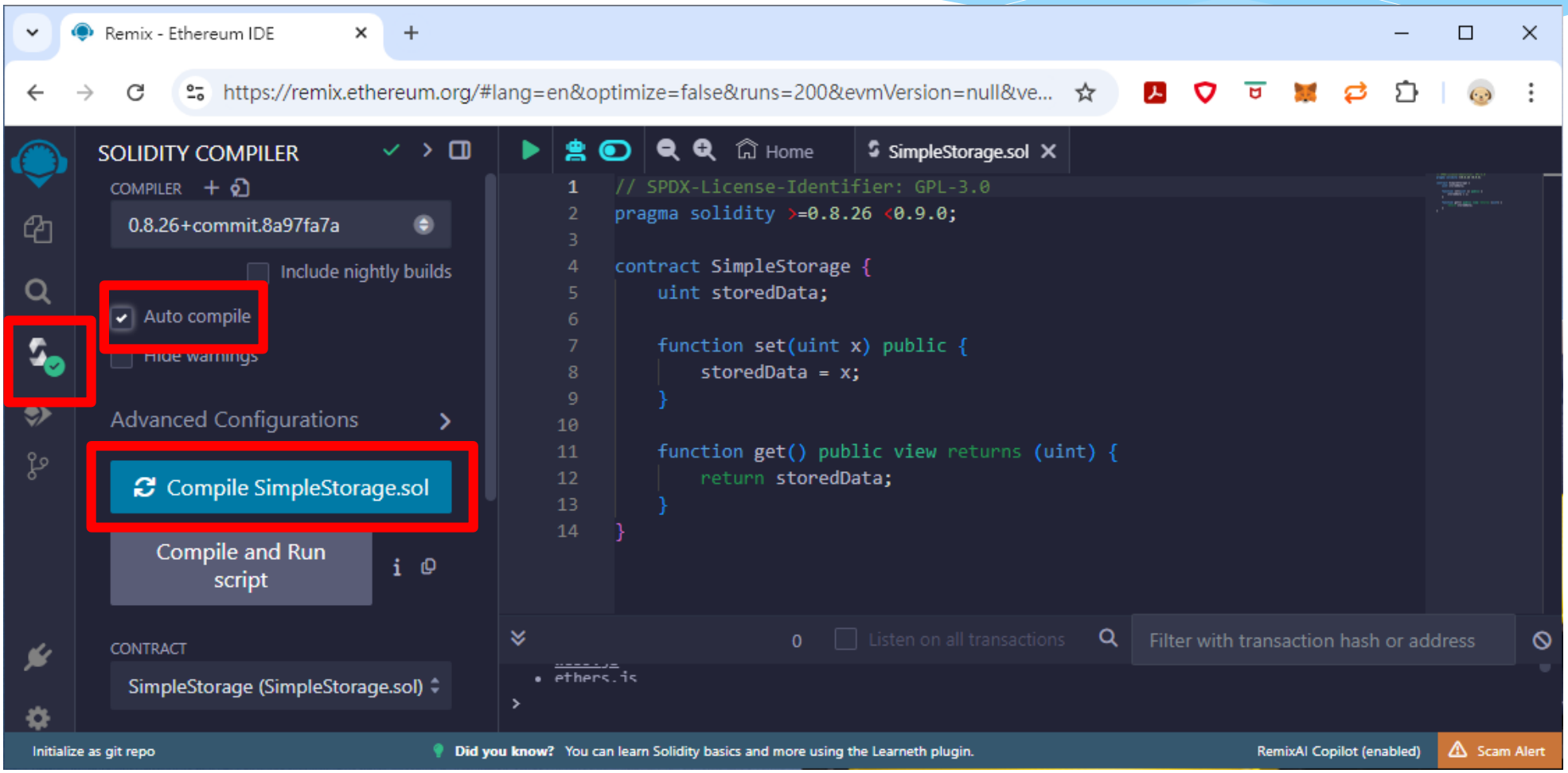
SimpleStorage.sol



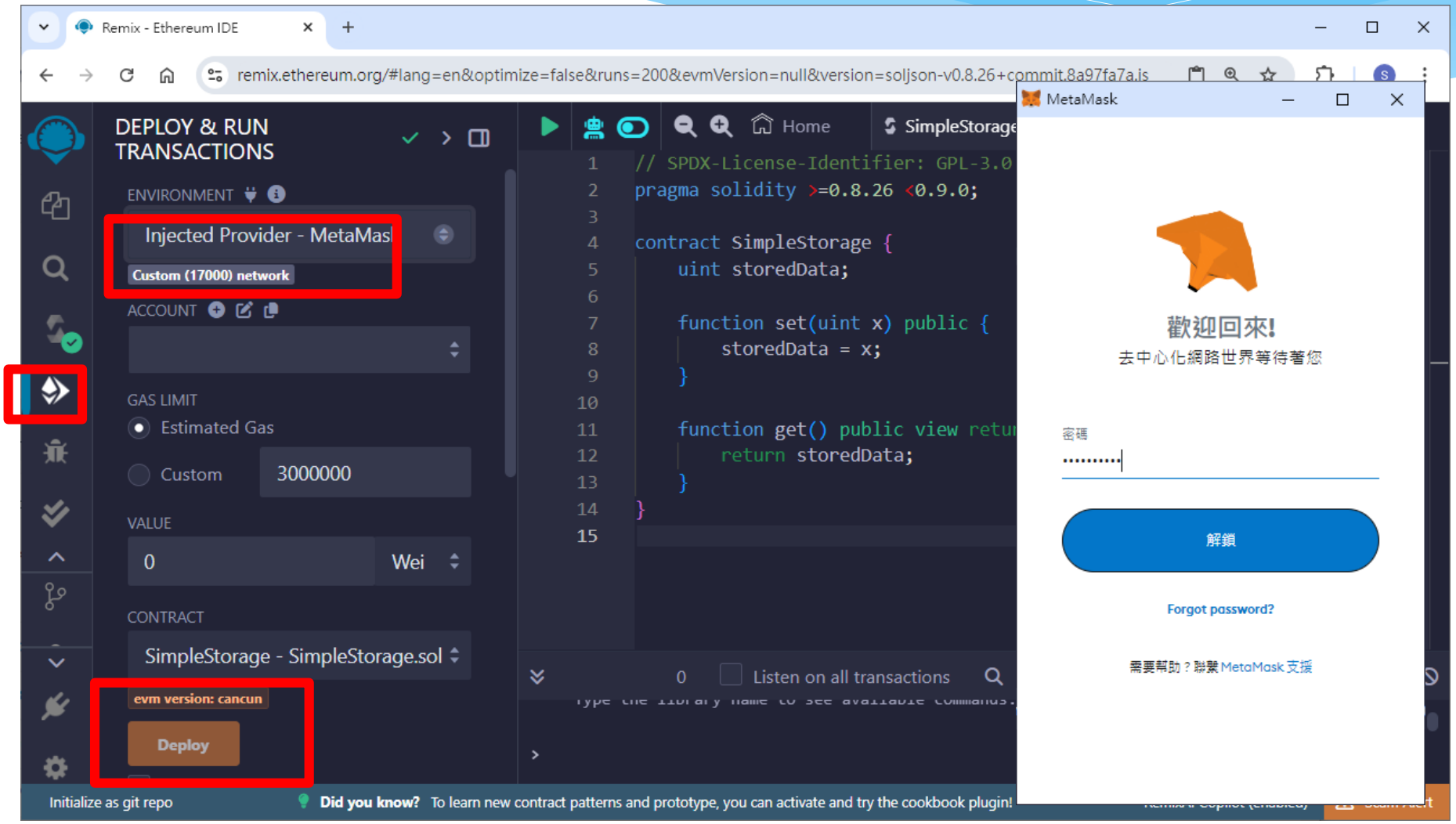
SimpleStorage.sol



Compile Smart Contract

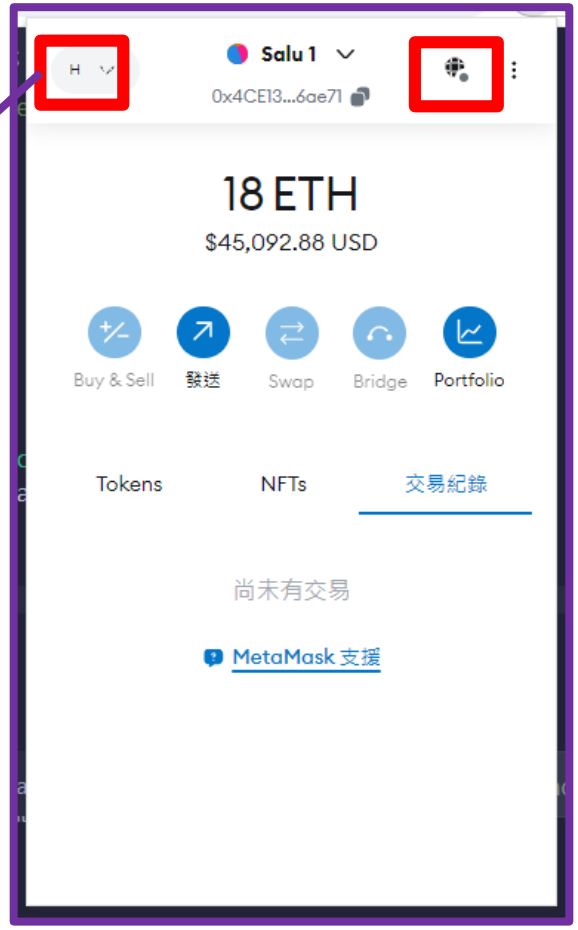


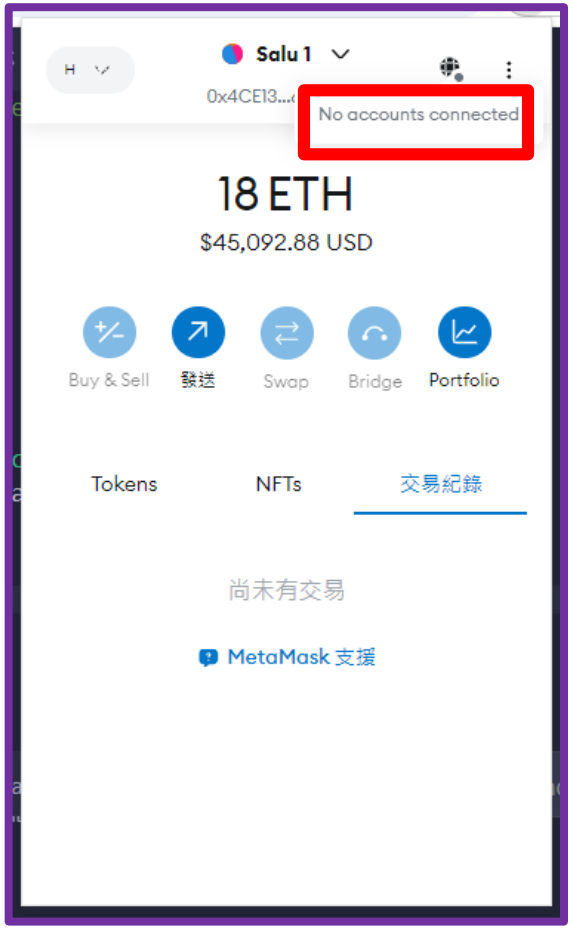
Deploy and Run Transaction



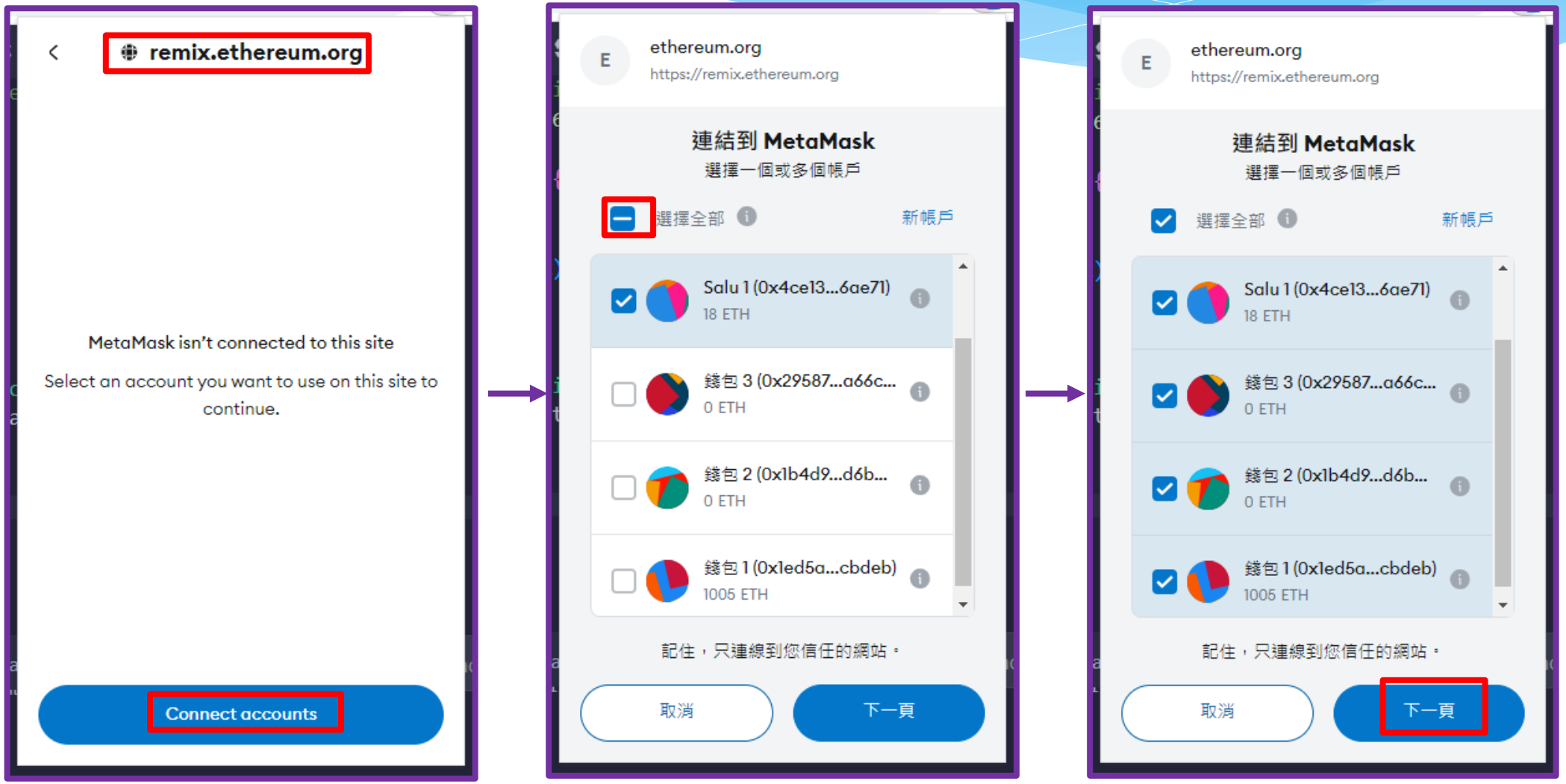
Connect to Holesky testnet

Holesky testnet

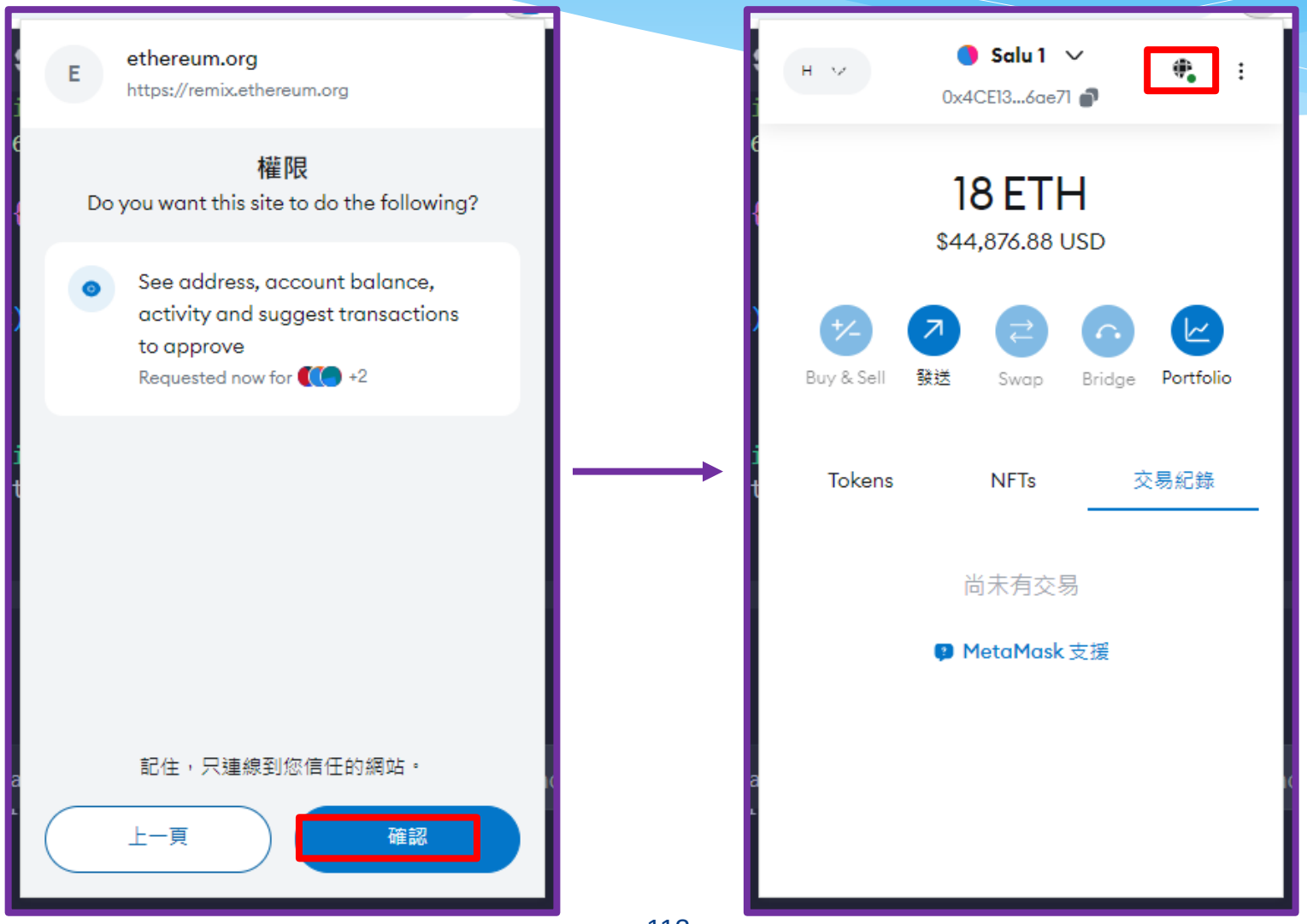




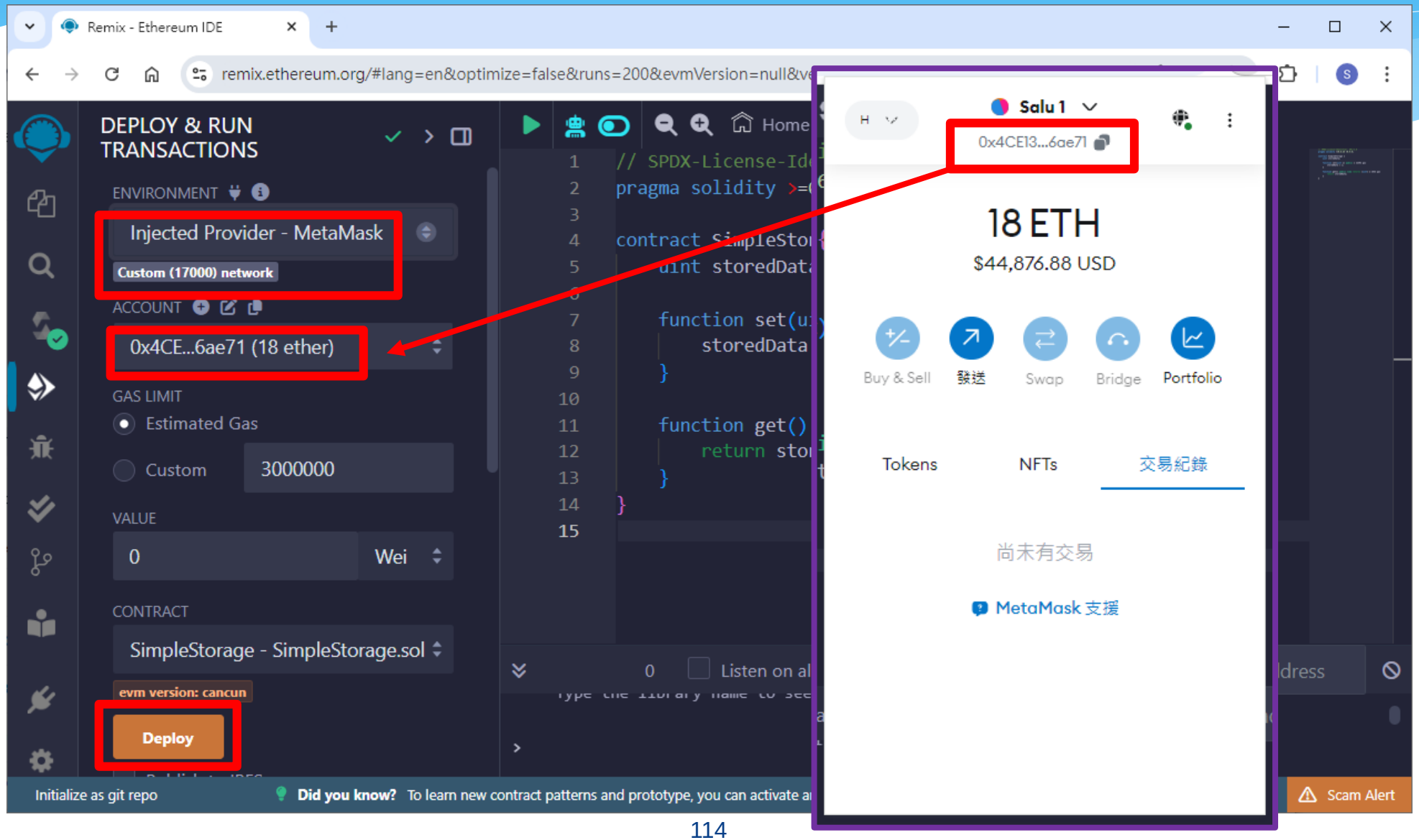
Connect to remix.ethereum.org



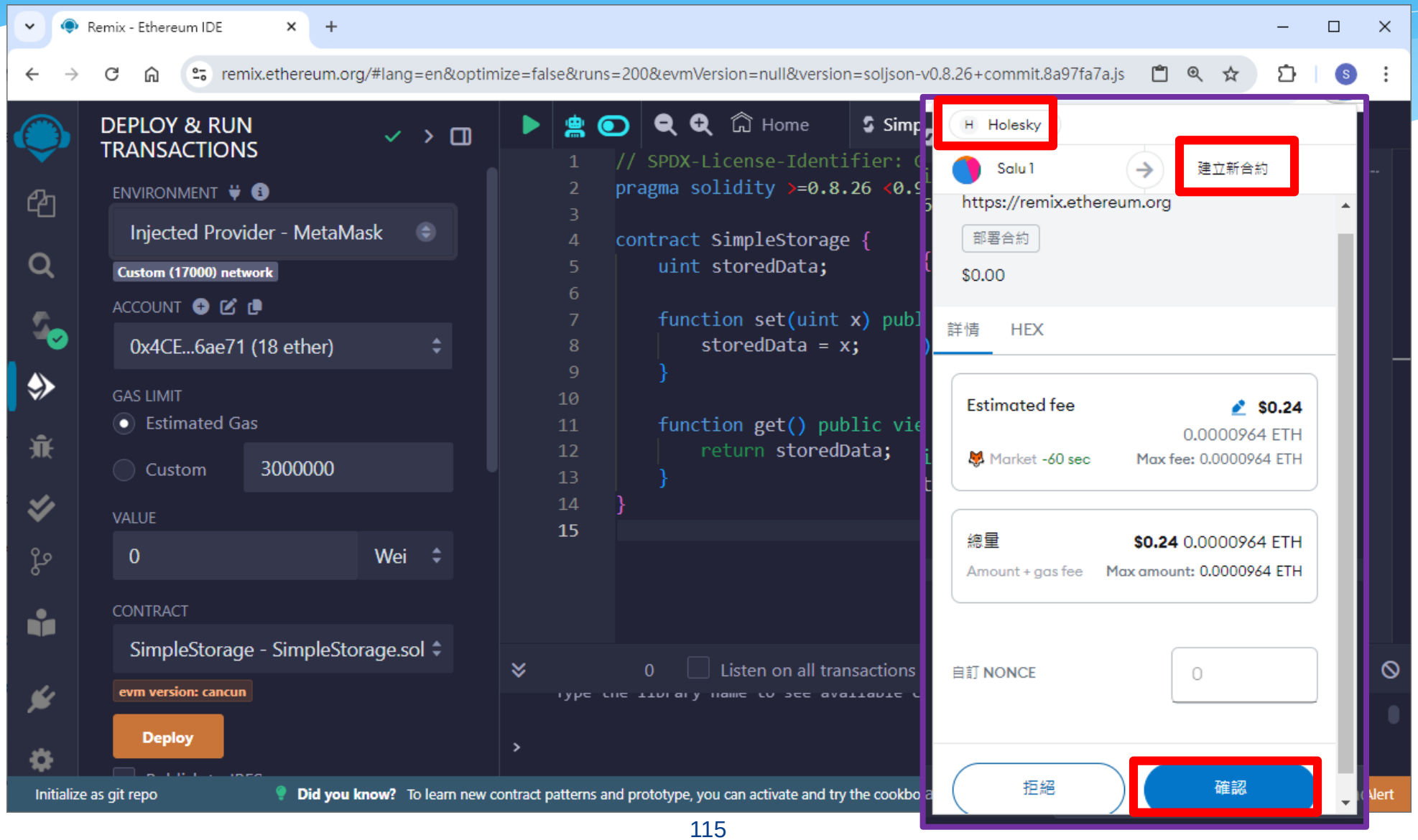
Connect to remix.ethereum.org



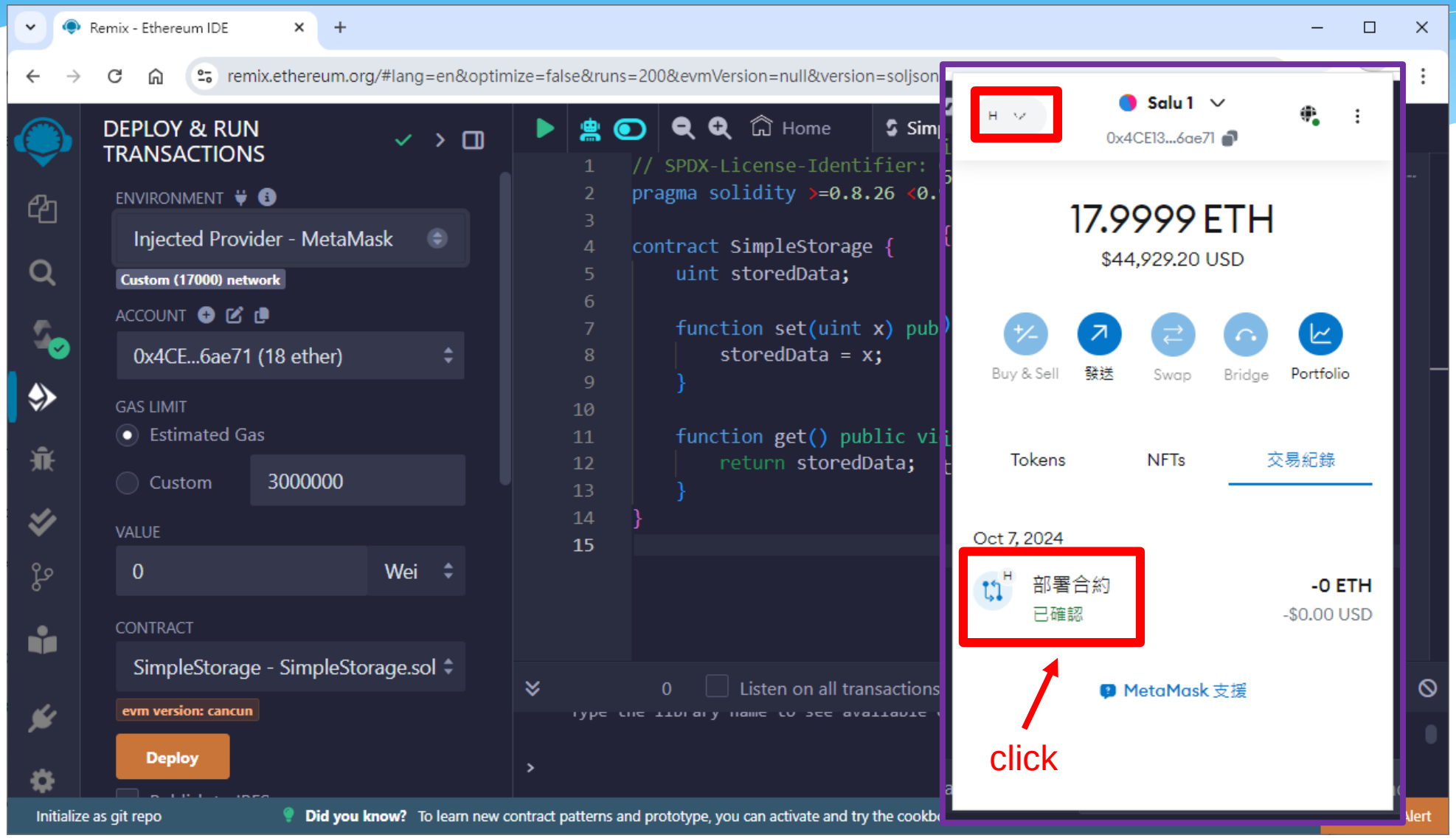
Deploy and Run Transaction



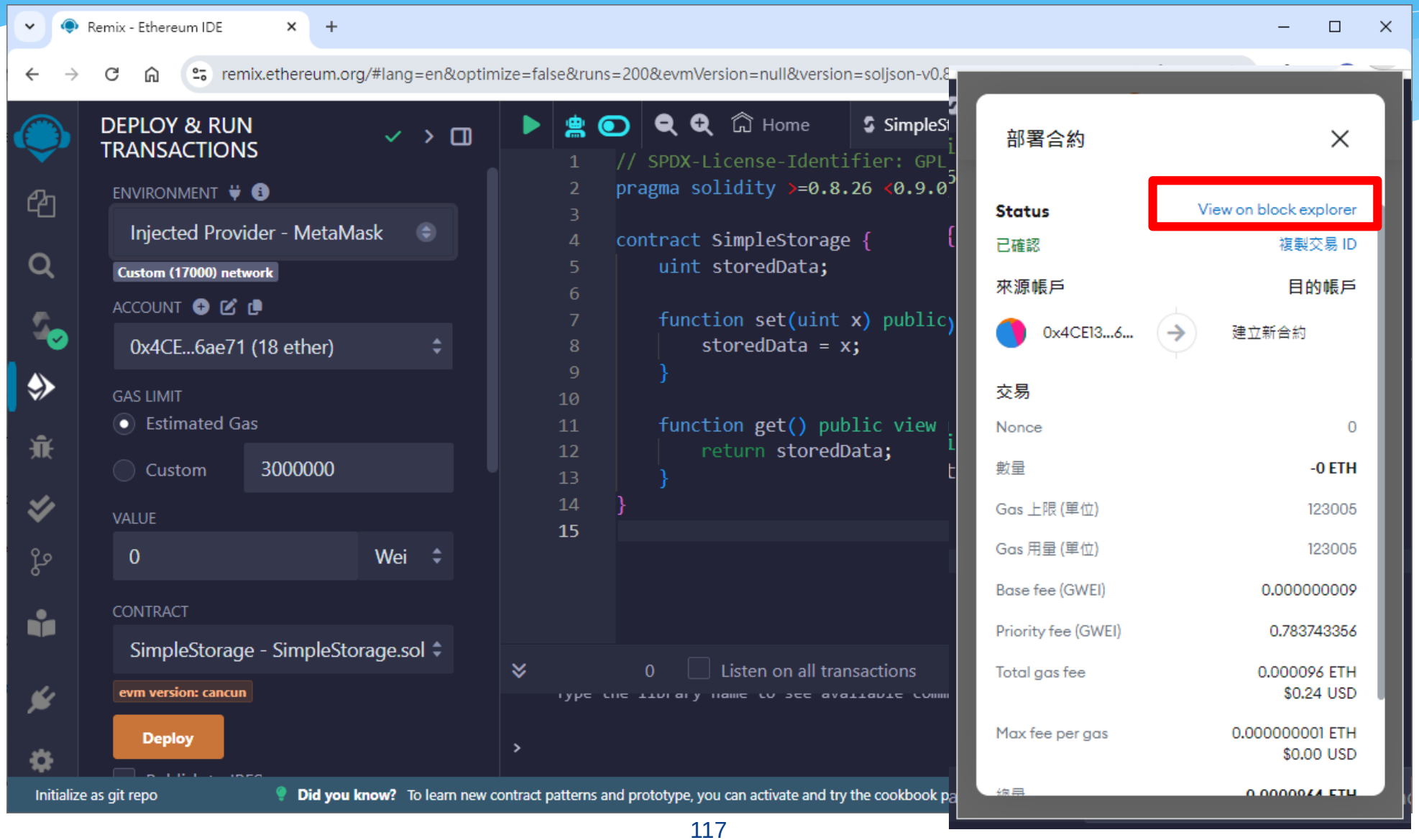
Deploy and Run Transaction



Deploy and Run Transaction

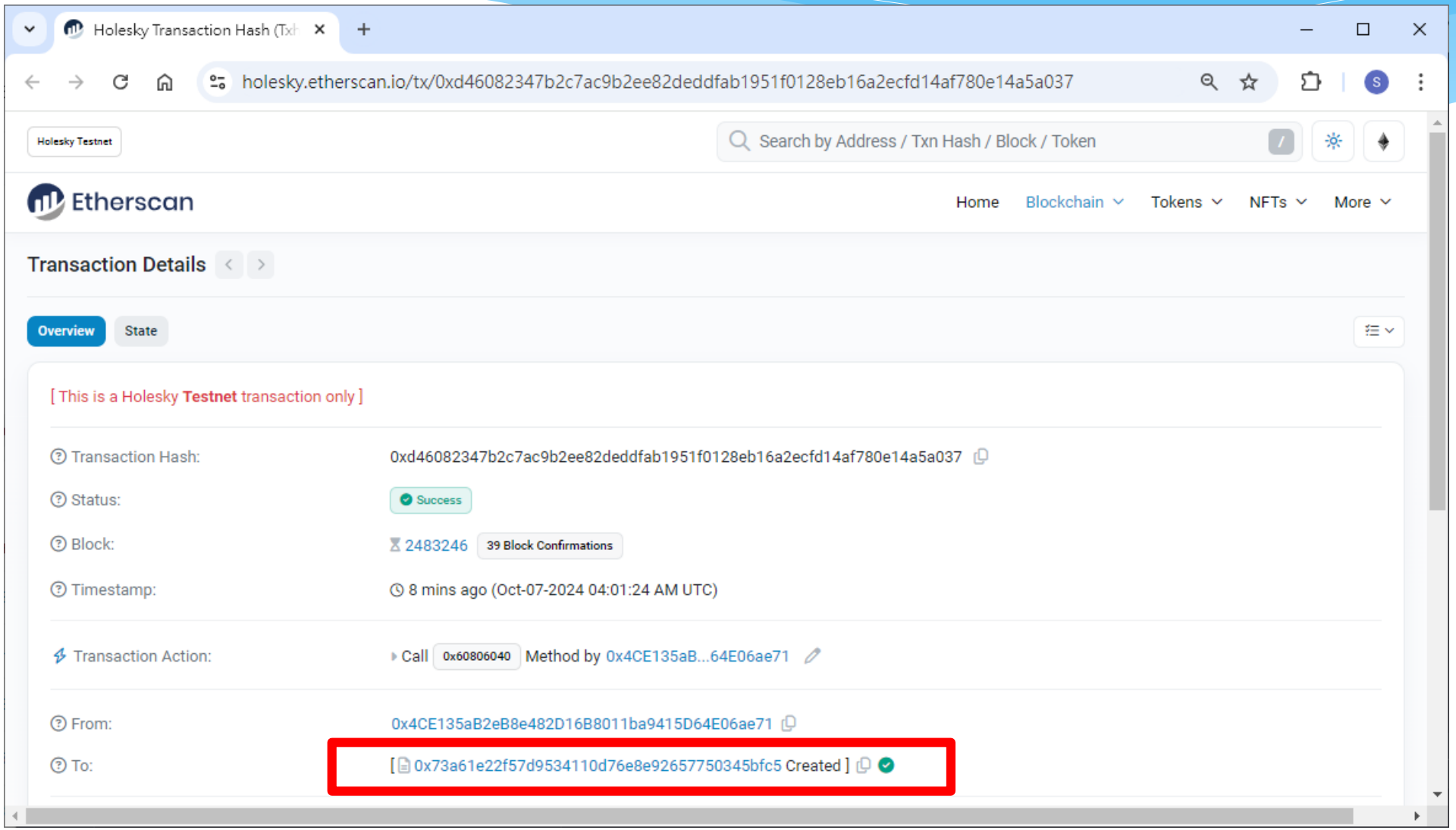


Deploy and Run Transaction



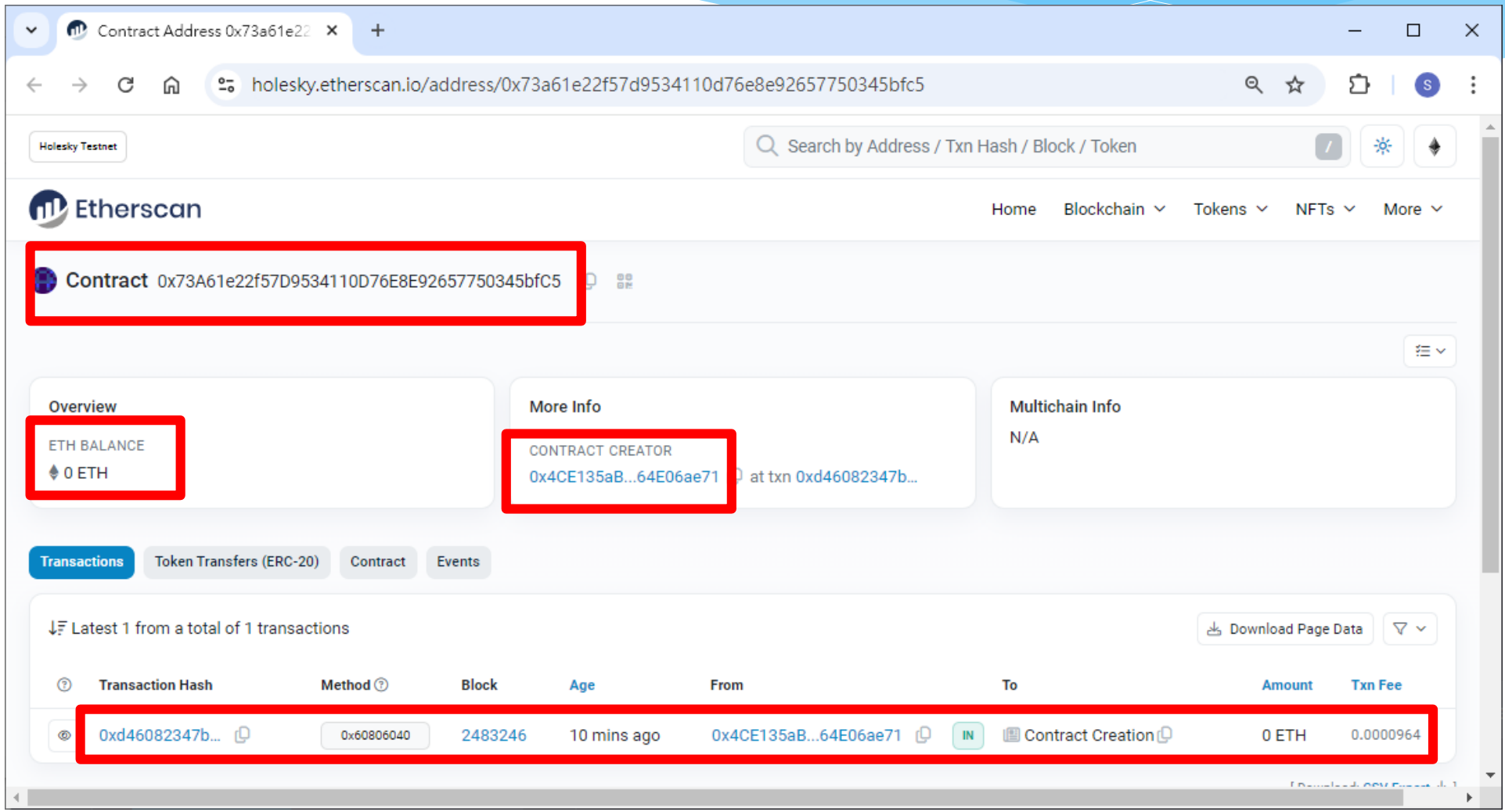
https://holesky.etherscan.io/tx/0xd46082347b2c7ac9b2ee82deddfab1951f0128eb16a2ecfd14af780e14a5a037

View Transaction on Holesky



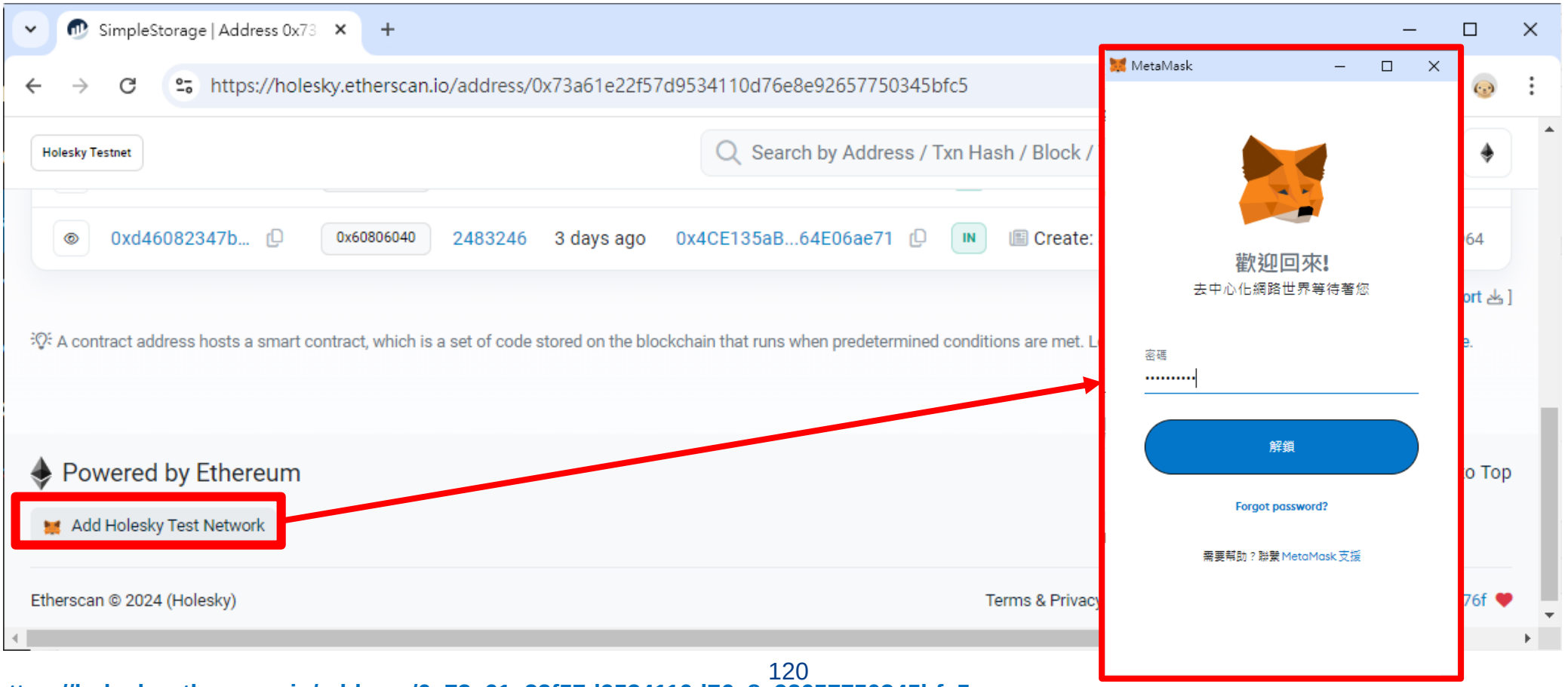
https://holesky.etherscan.io/tx/0xd46082347b2c7ac9b2ee82deddfab1951f0128eb16a2ecfd14af780e14a5a037

Smart Contract Information



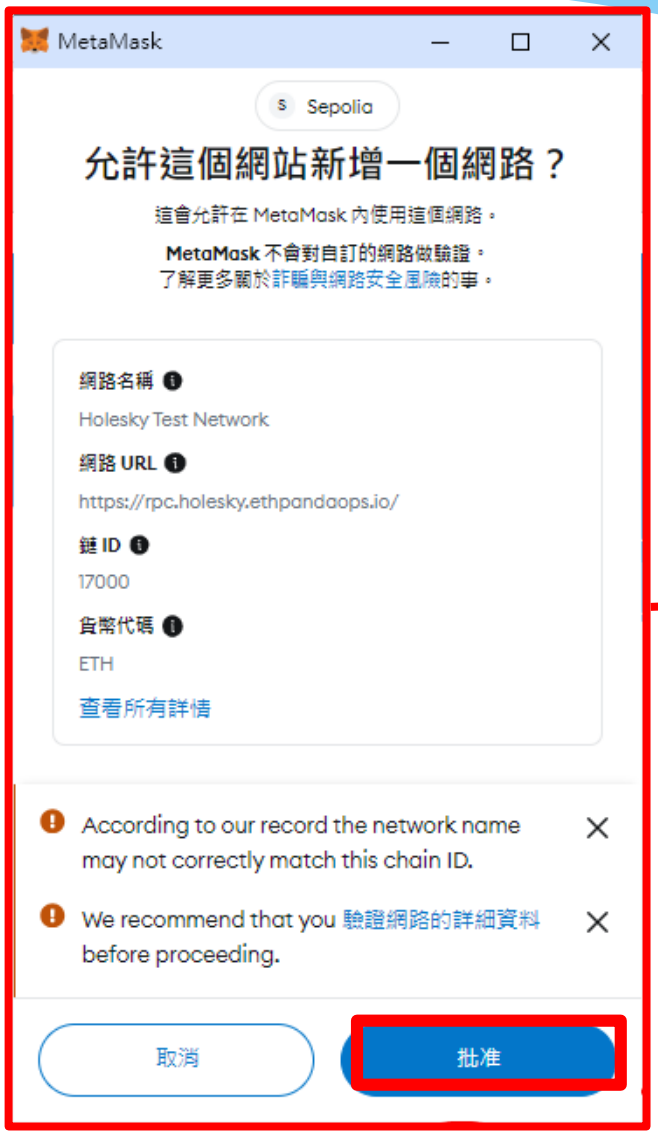
https://holesky.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5

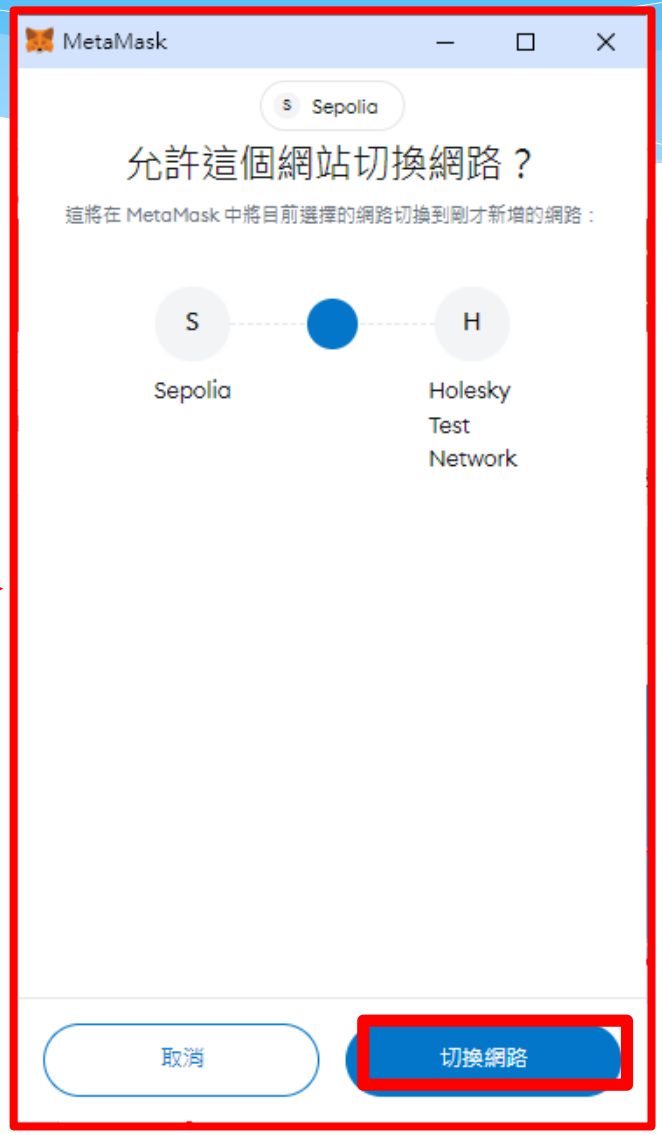
Add Holesky Test Network



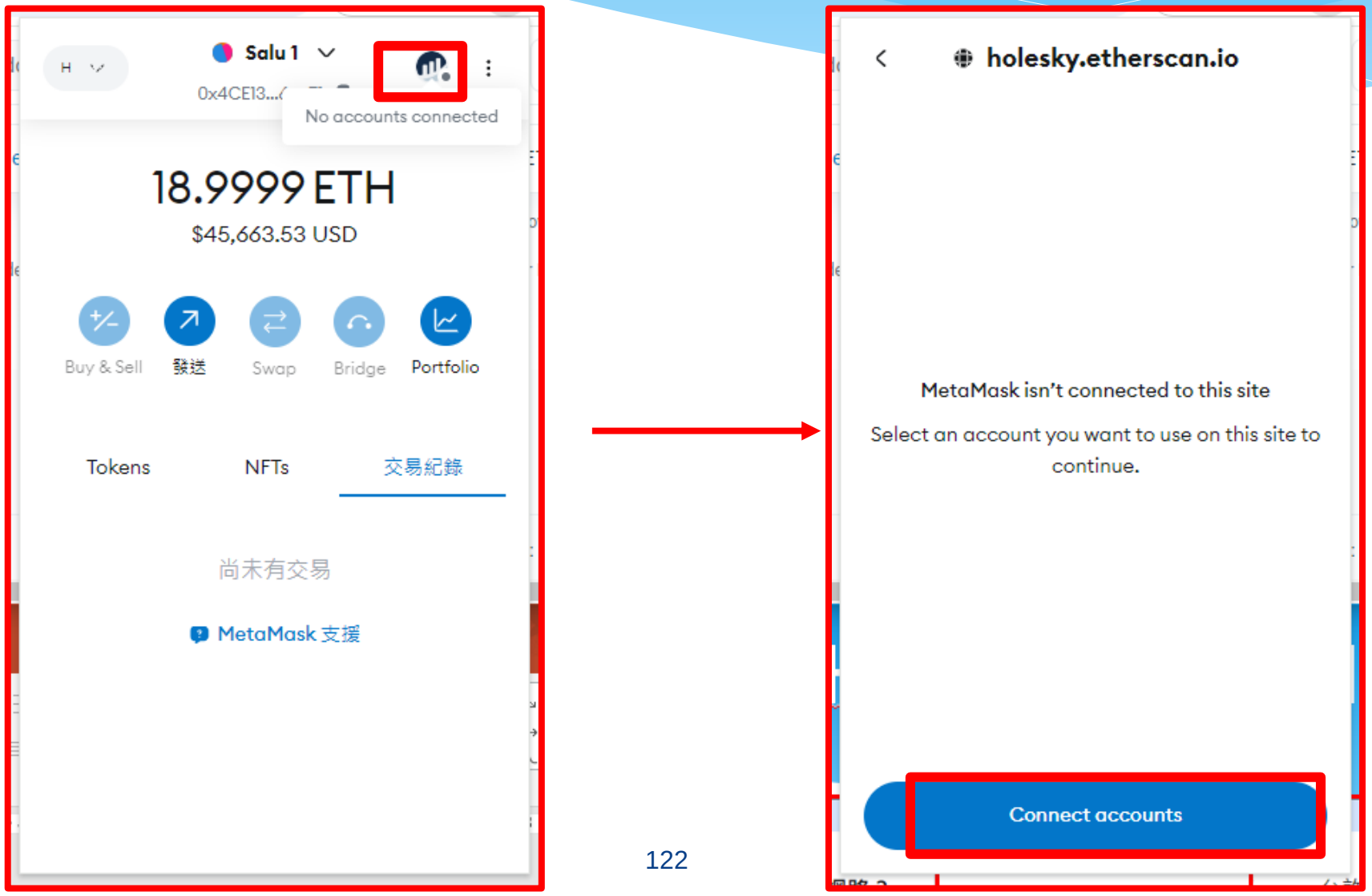
https://holesky.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5

Add Holesky Test Network

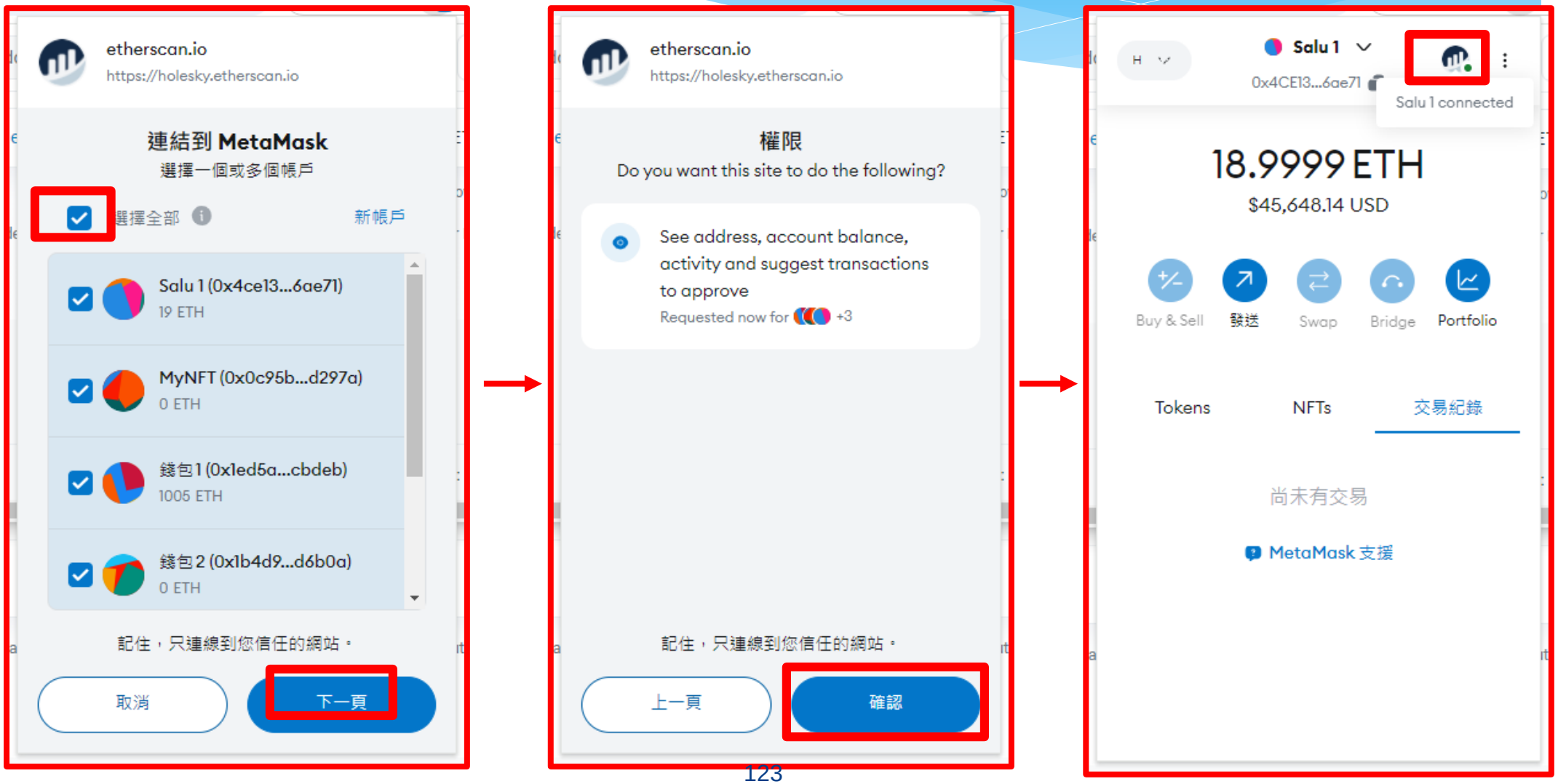




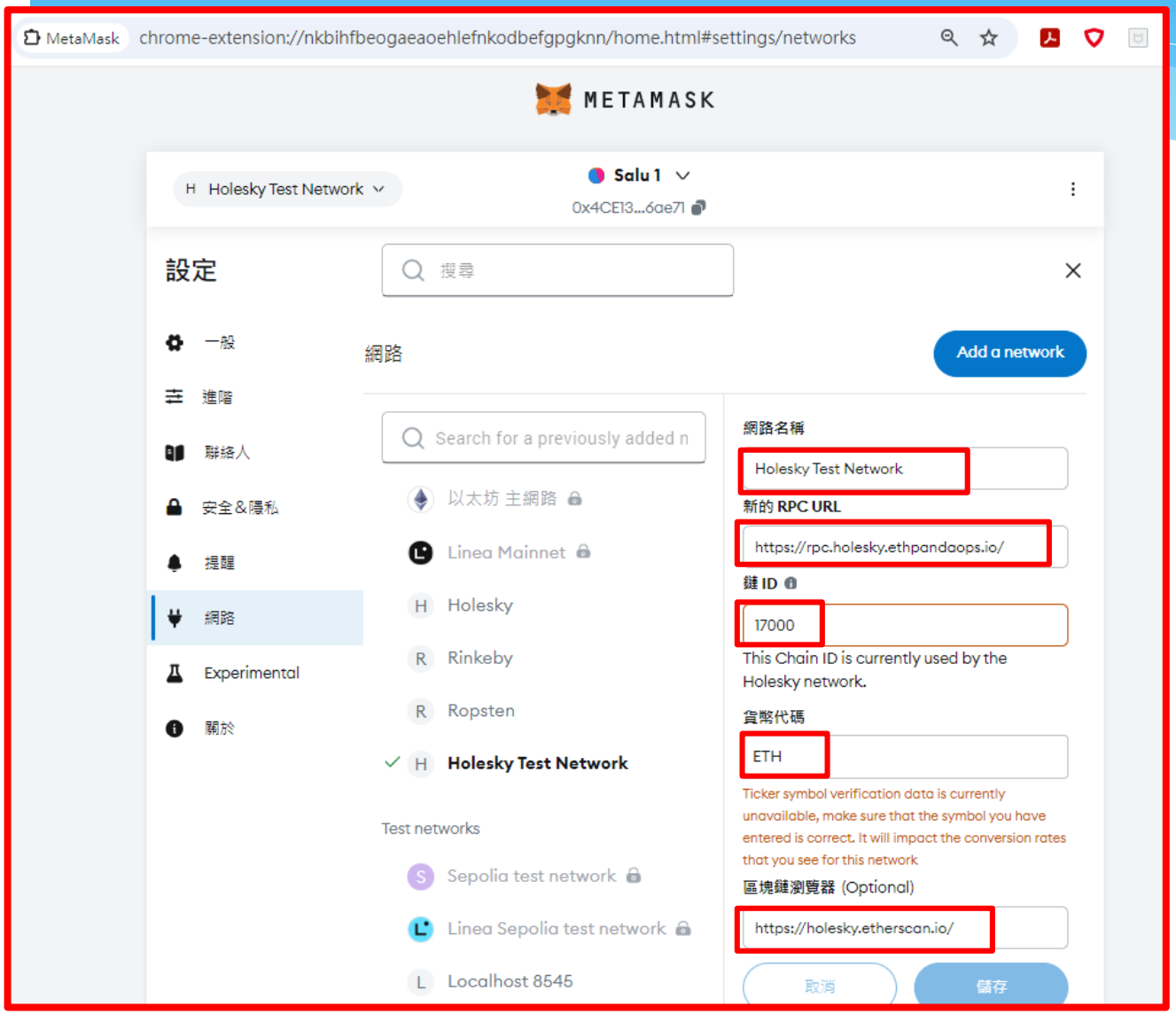
Add Holesky Test Network



Add Holesky Test Network



Add Holesky Test Network



網路名稱：Holesky Test Network

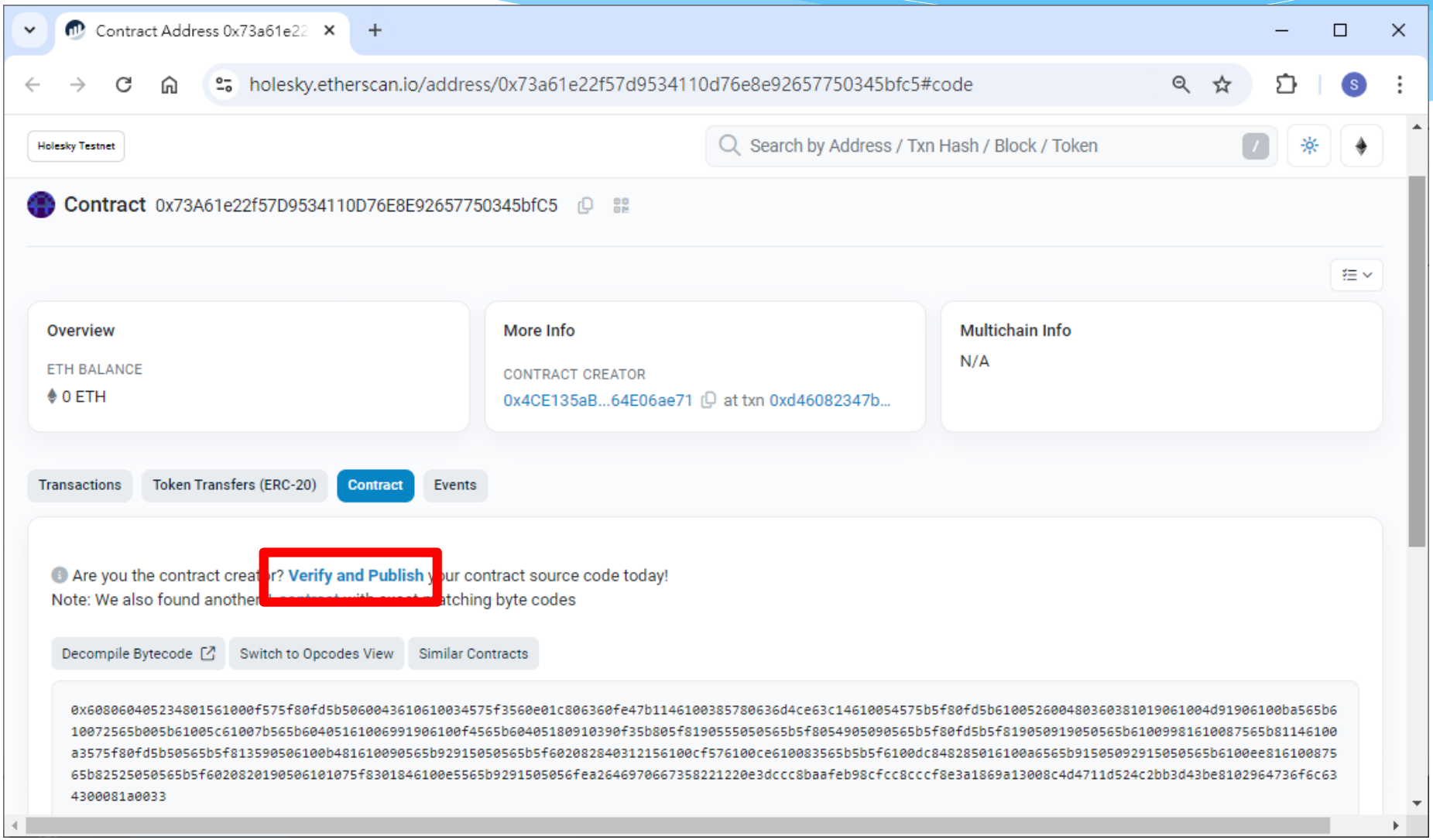
新的 RPC URL：https://rpc.holesky.ethpandaops.io/

鏈 ID：17000

貨幣代碼：ETH

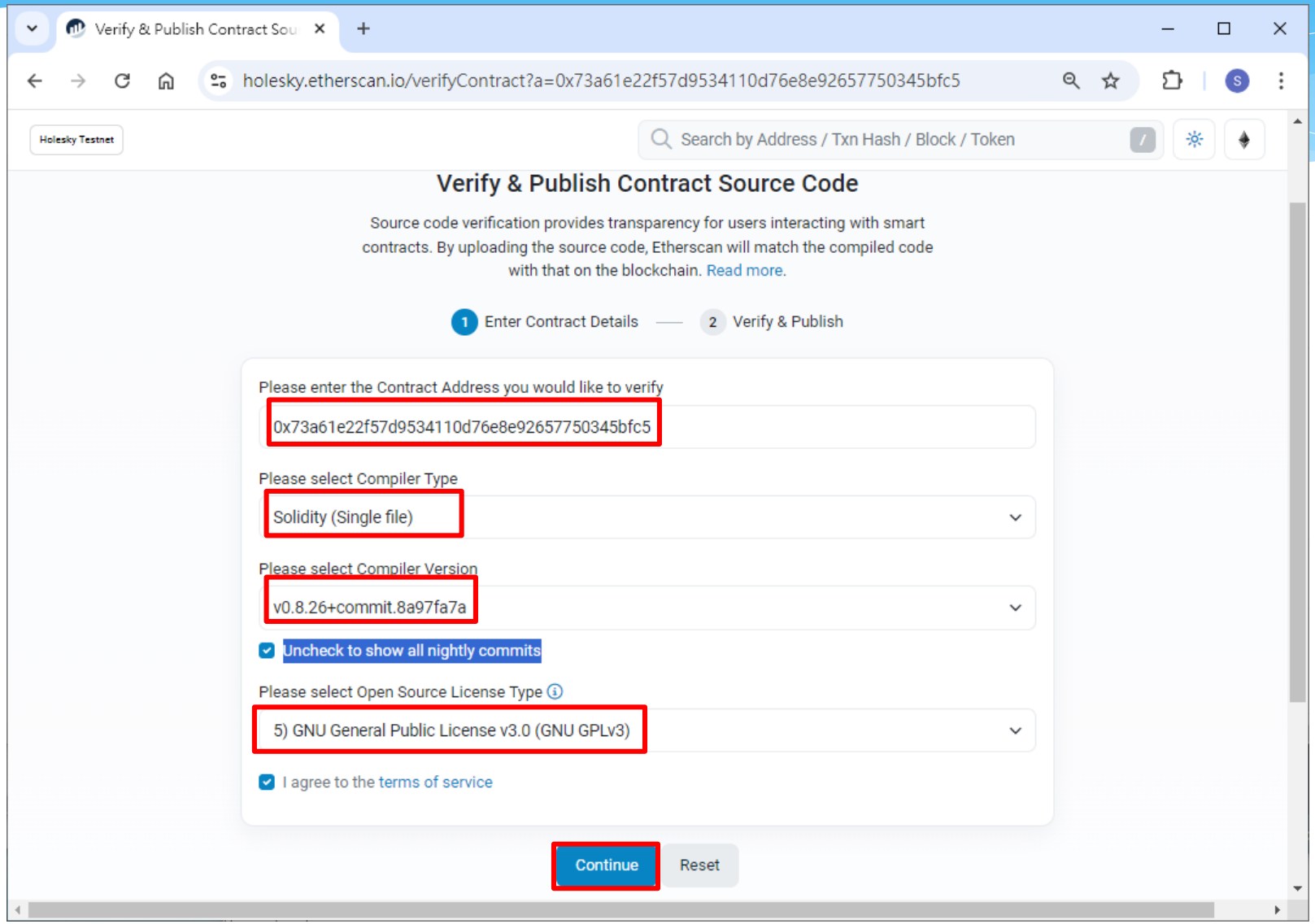
區塊鏈瀏覽器：https://holesky.etherscan.io/

View & Publish



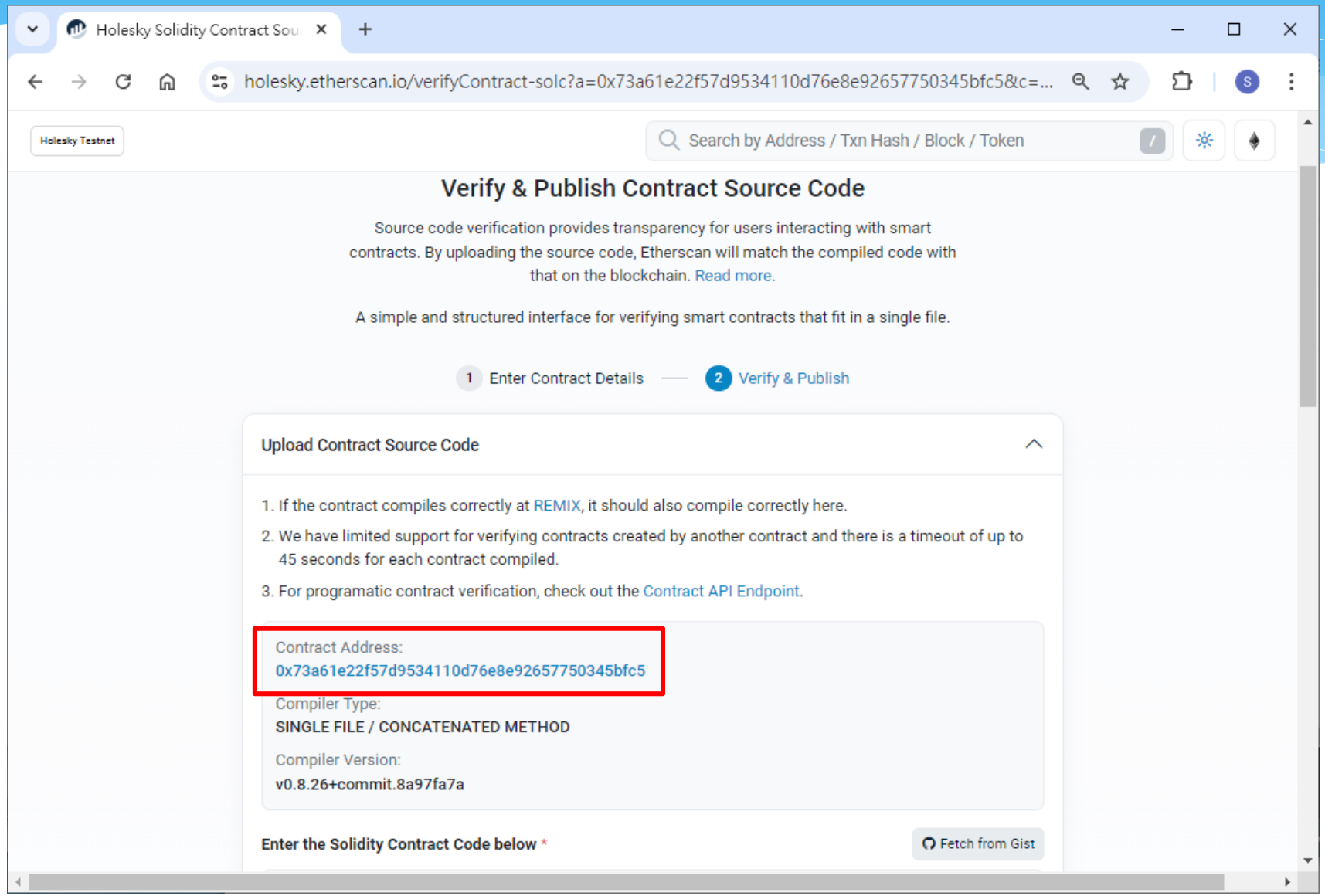
https://holesky.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5#code

View & Publish Contract Source Code

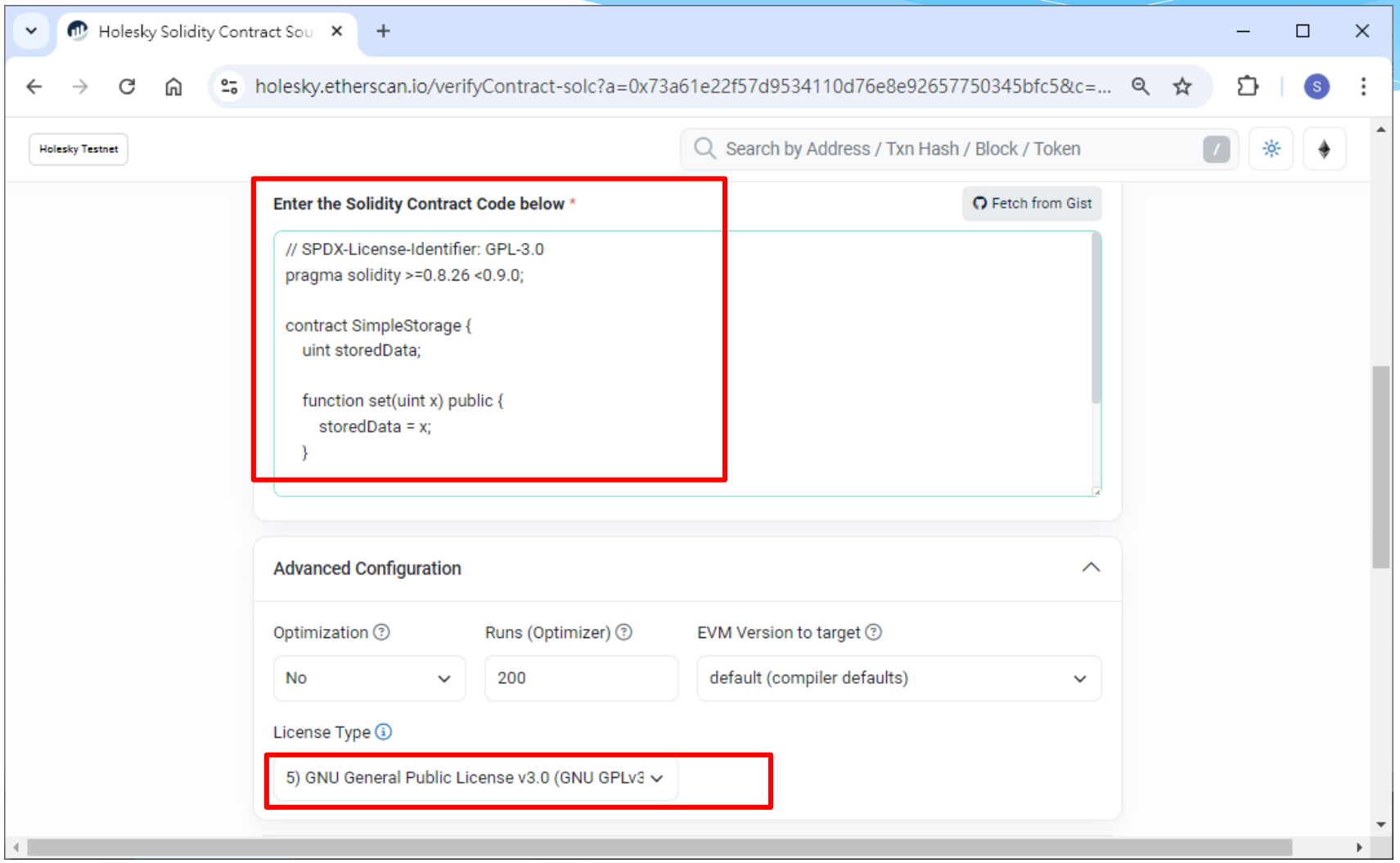


https://holesky.etherscan.io/verifyContract?a=0x73a61e22f57d9534110d76e8e92657750345bfc5

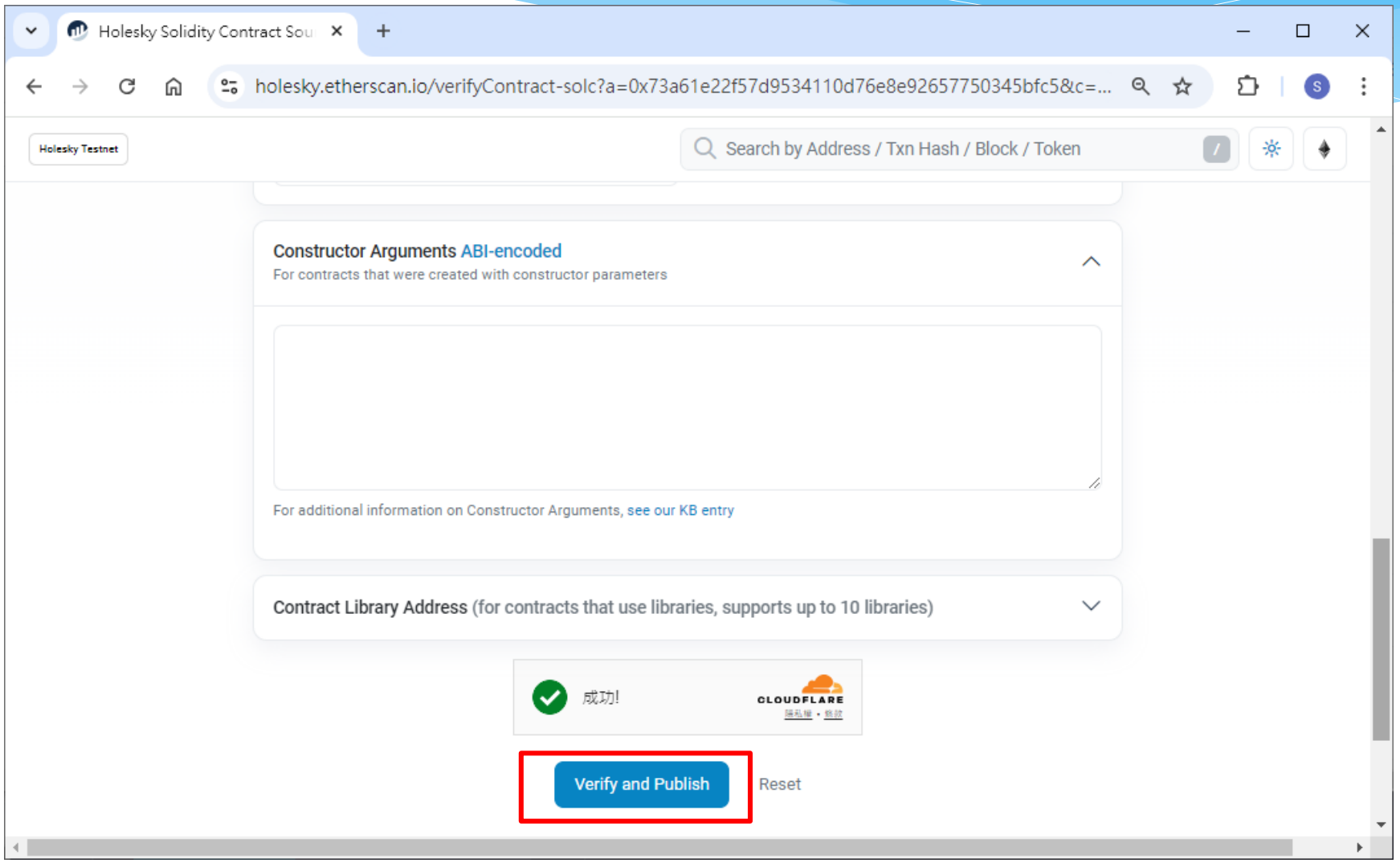
View & Publish Contract Source Code



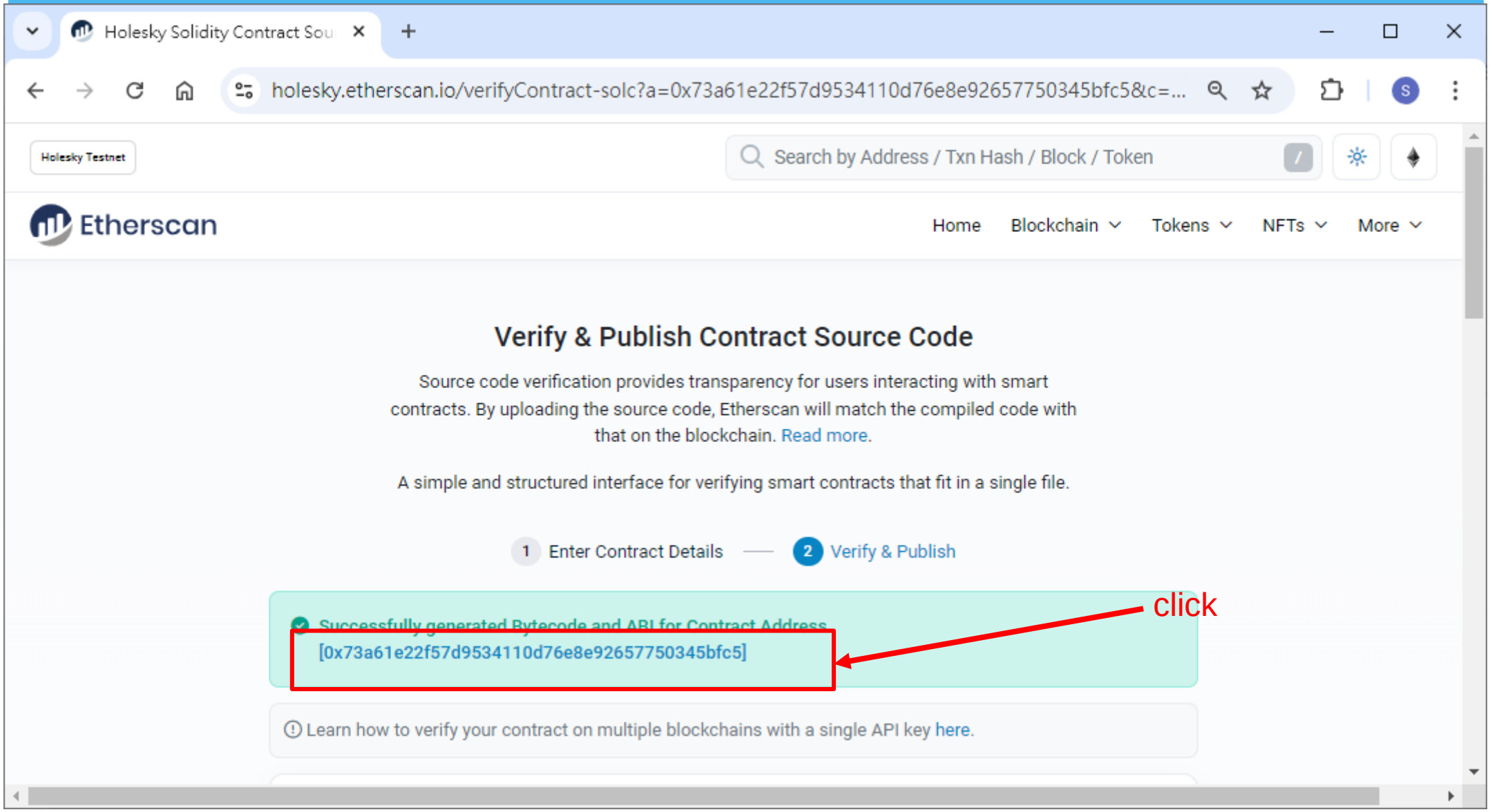
View & Publish Contract Source Code



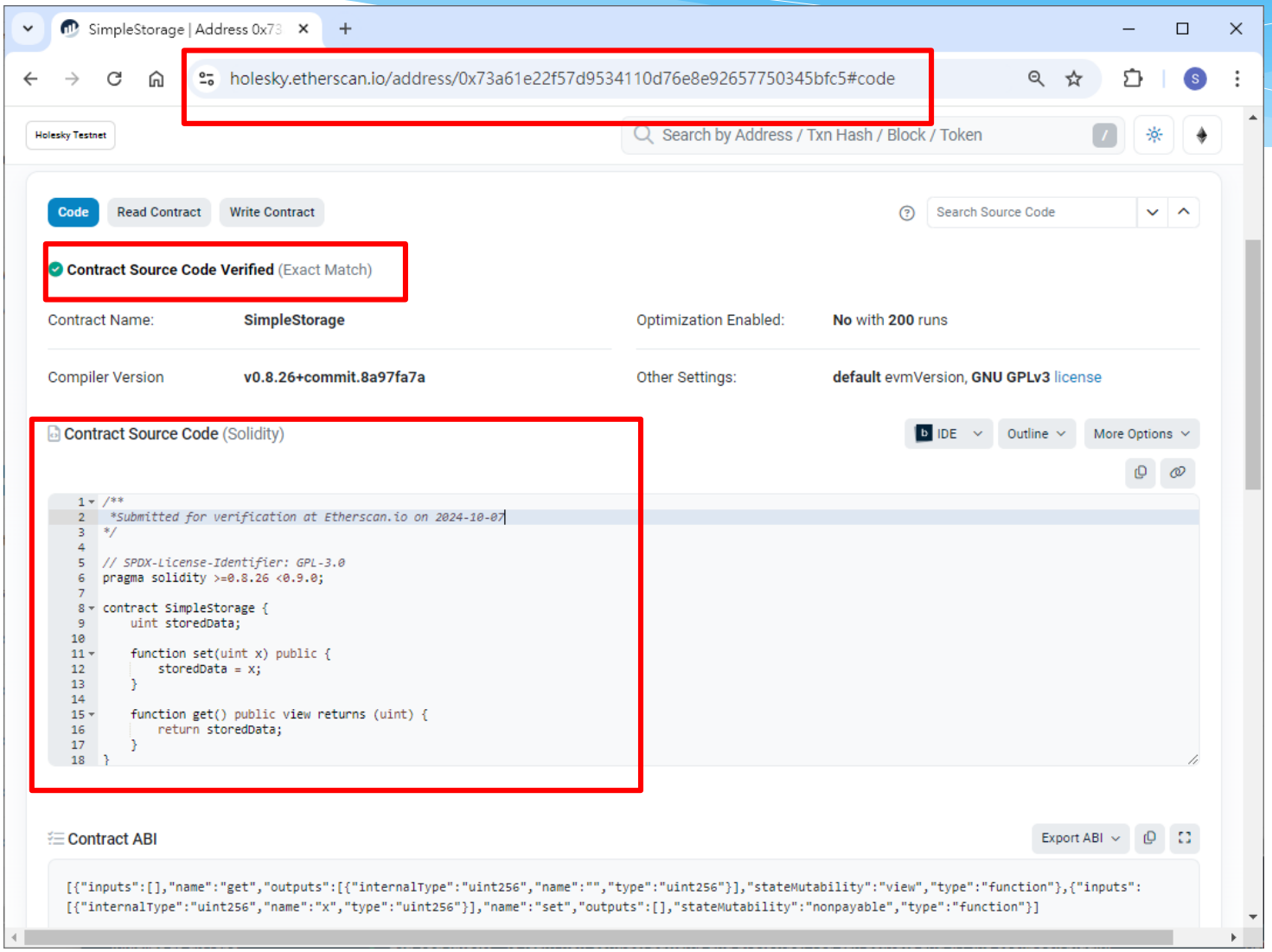
View & Publish Contract Source Code



View & Publish Contract Source Code

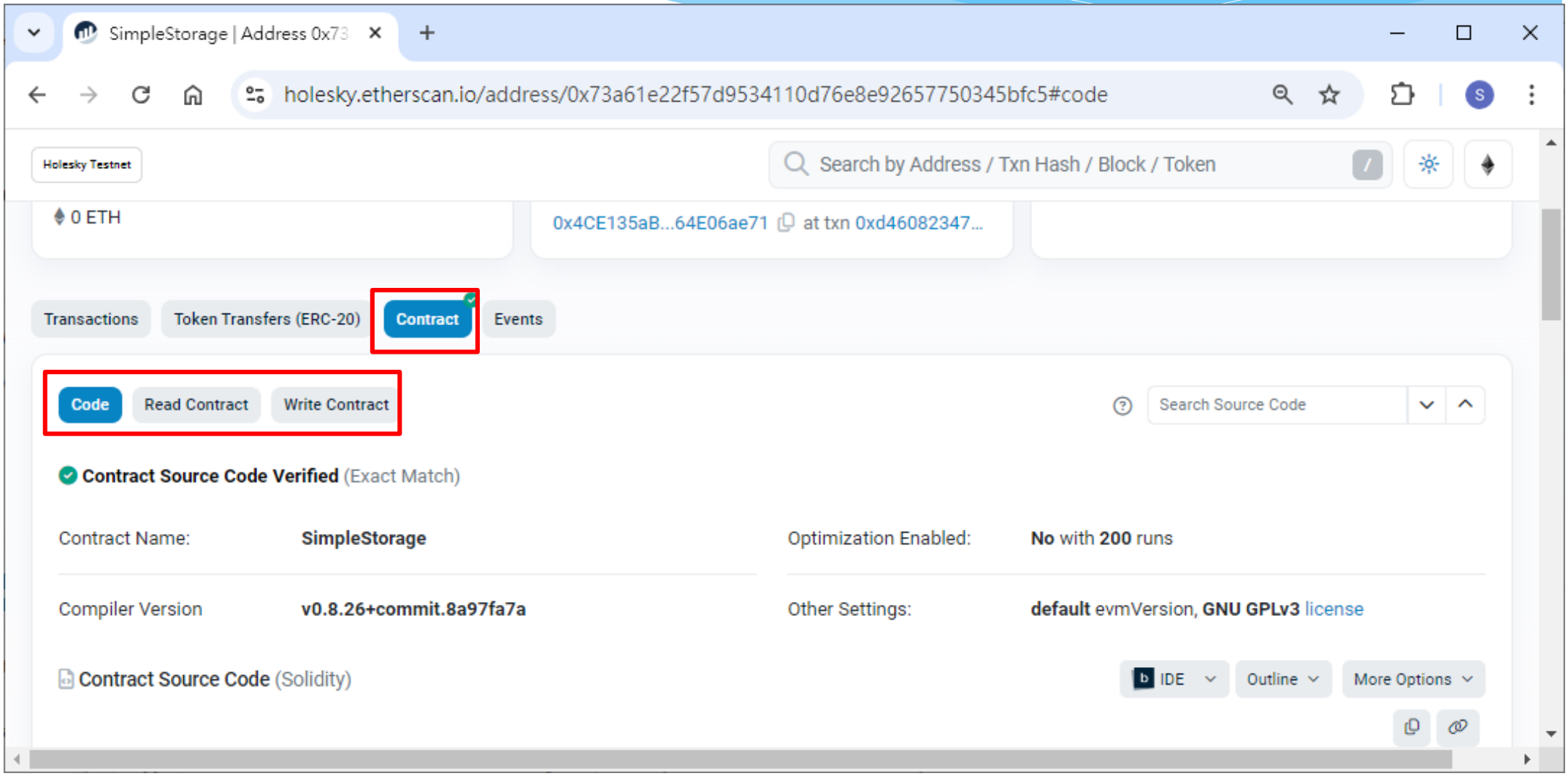


View Contract Source Code



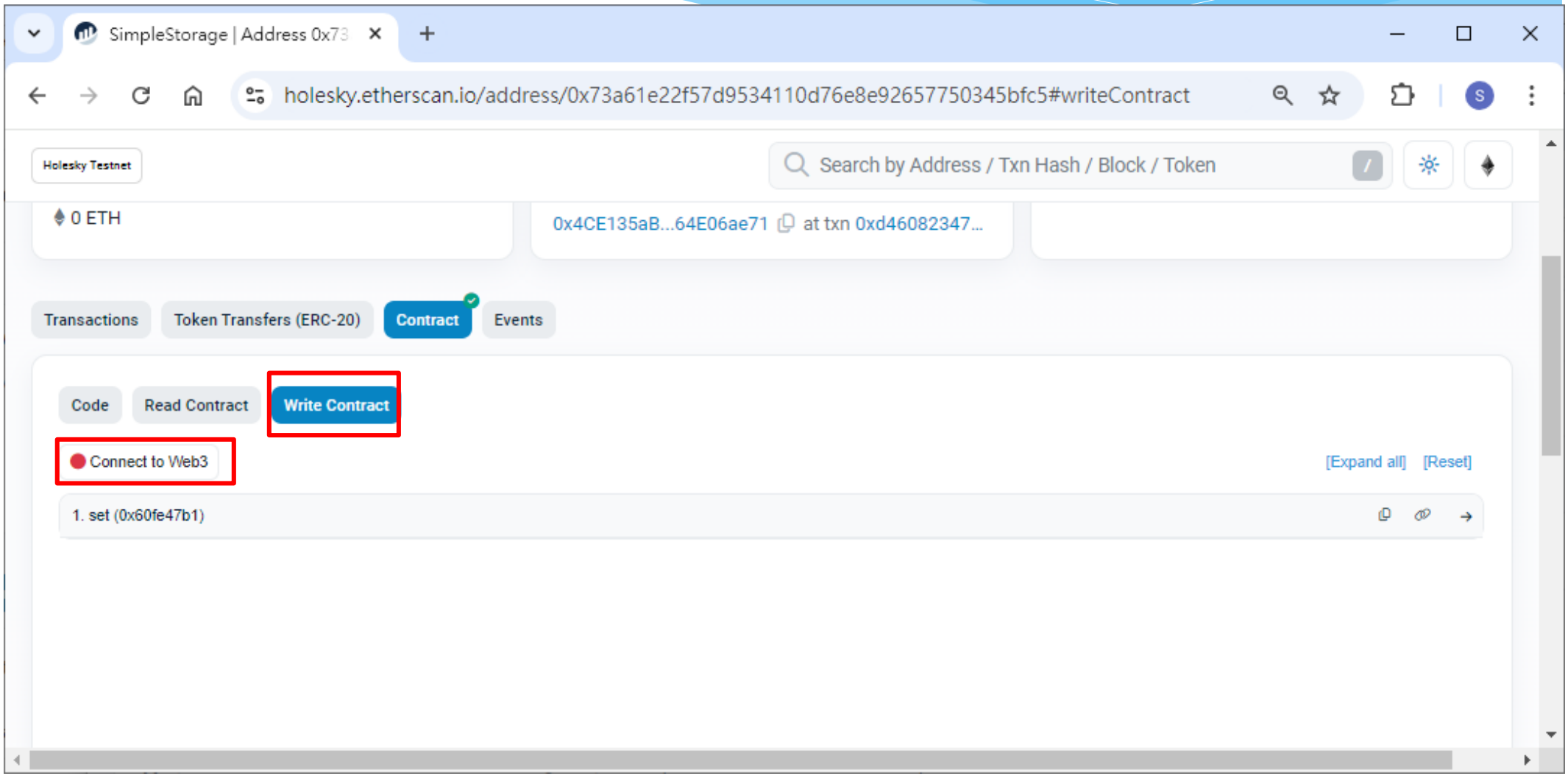
https://holesky.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5#code

View & Publish Contract Source Code



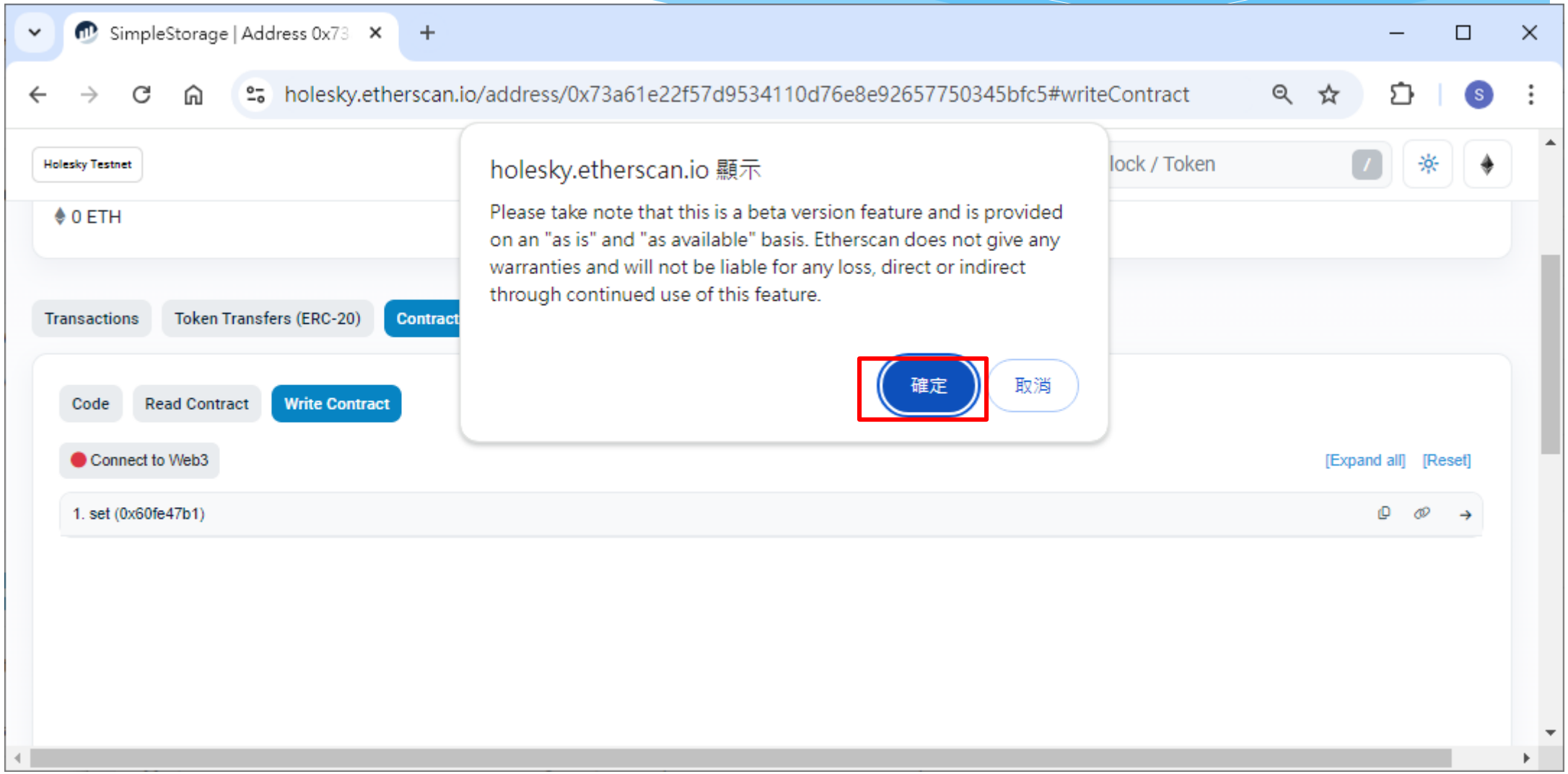
https://holesky.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5#code

Write Contract-Connected Web3



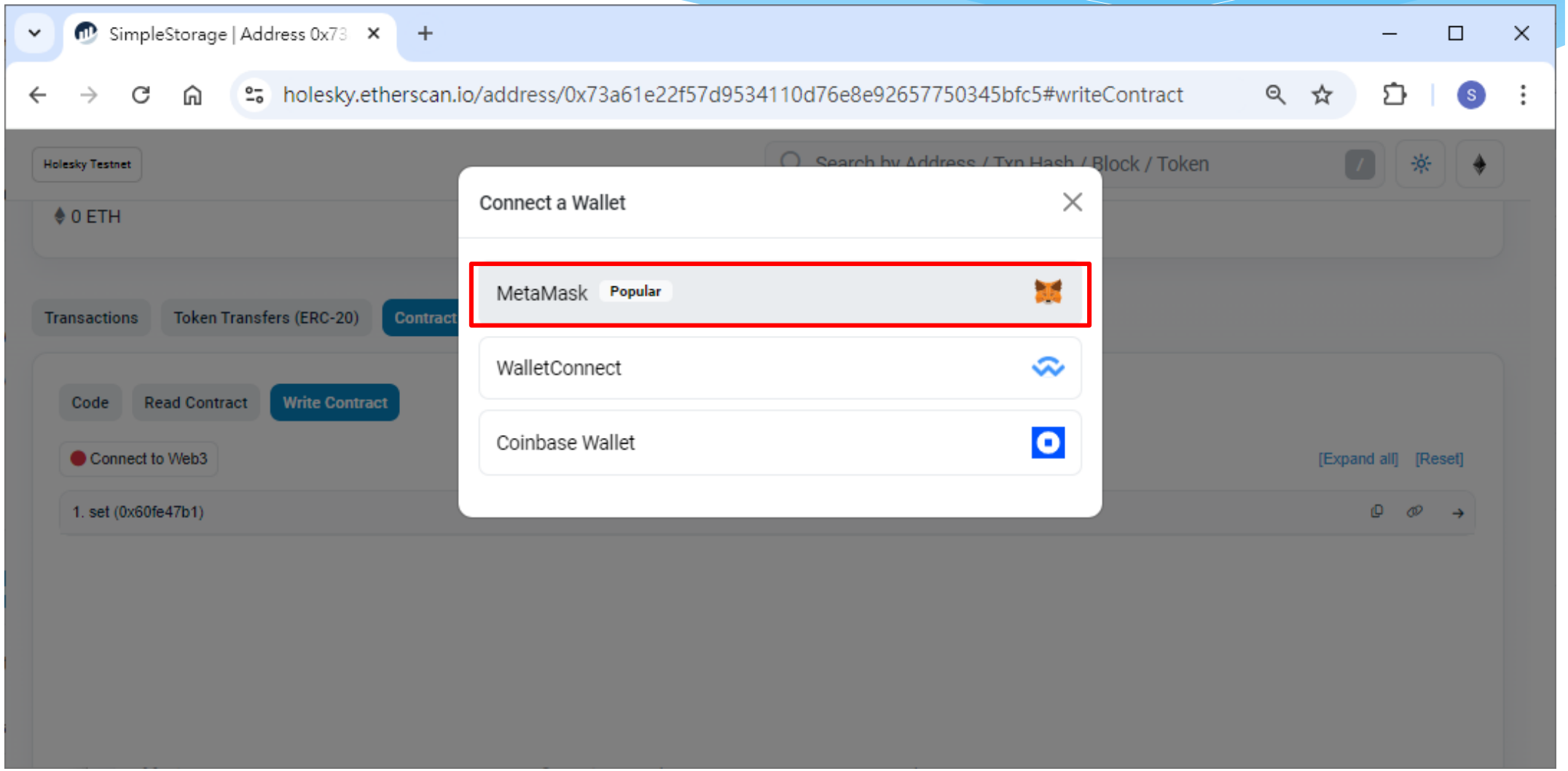
https://holesky.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5#writeContract

Connect to Web3



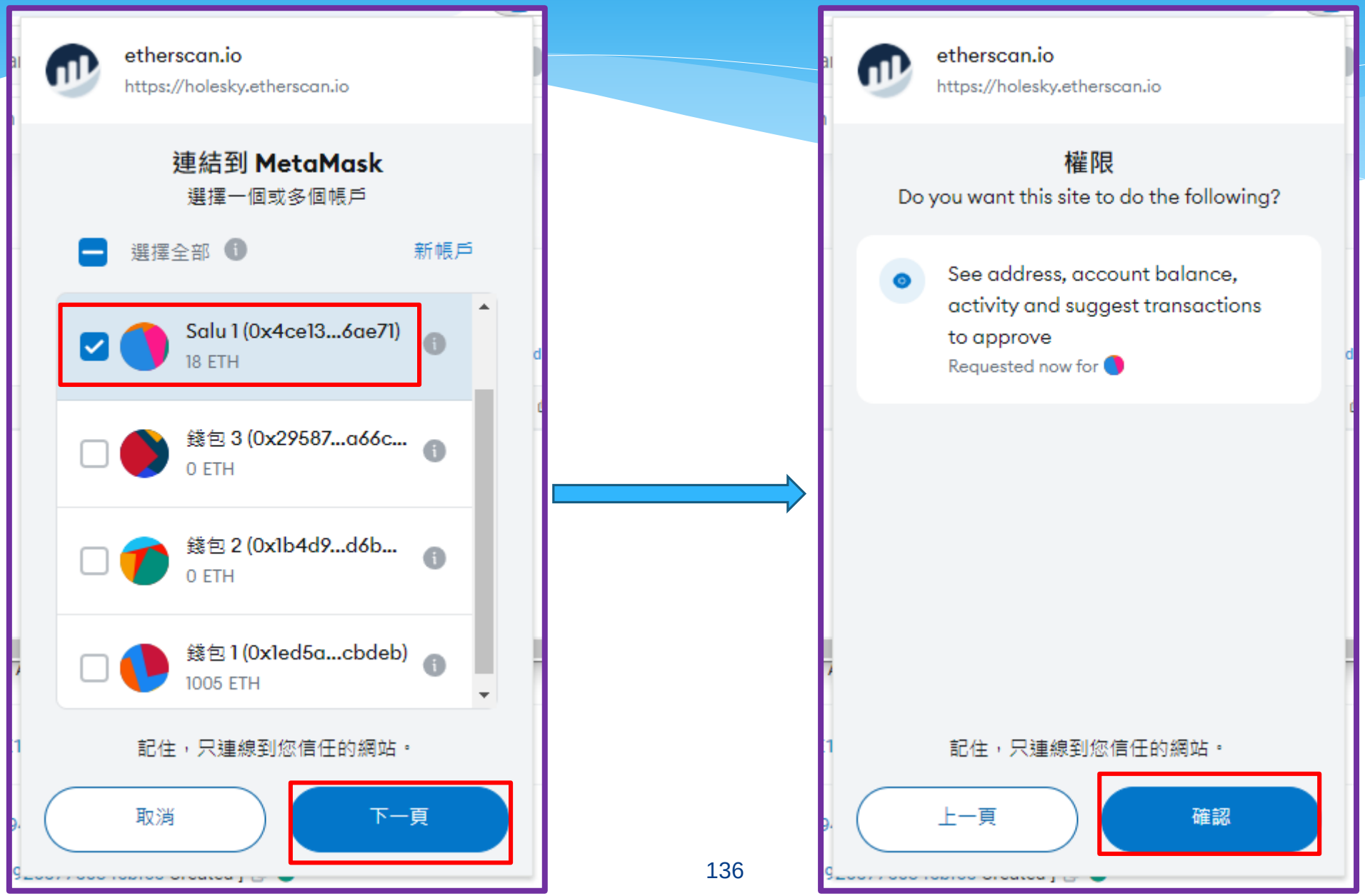
https://holesky.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5#writeContract

Connect to Web3

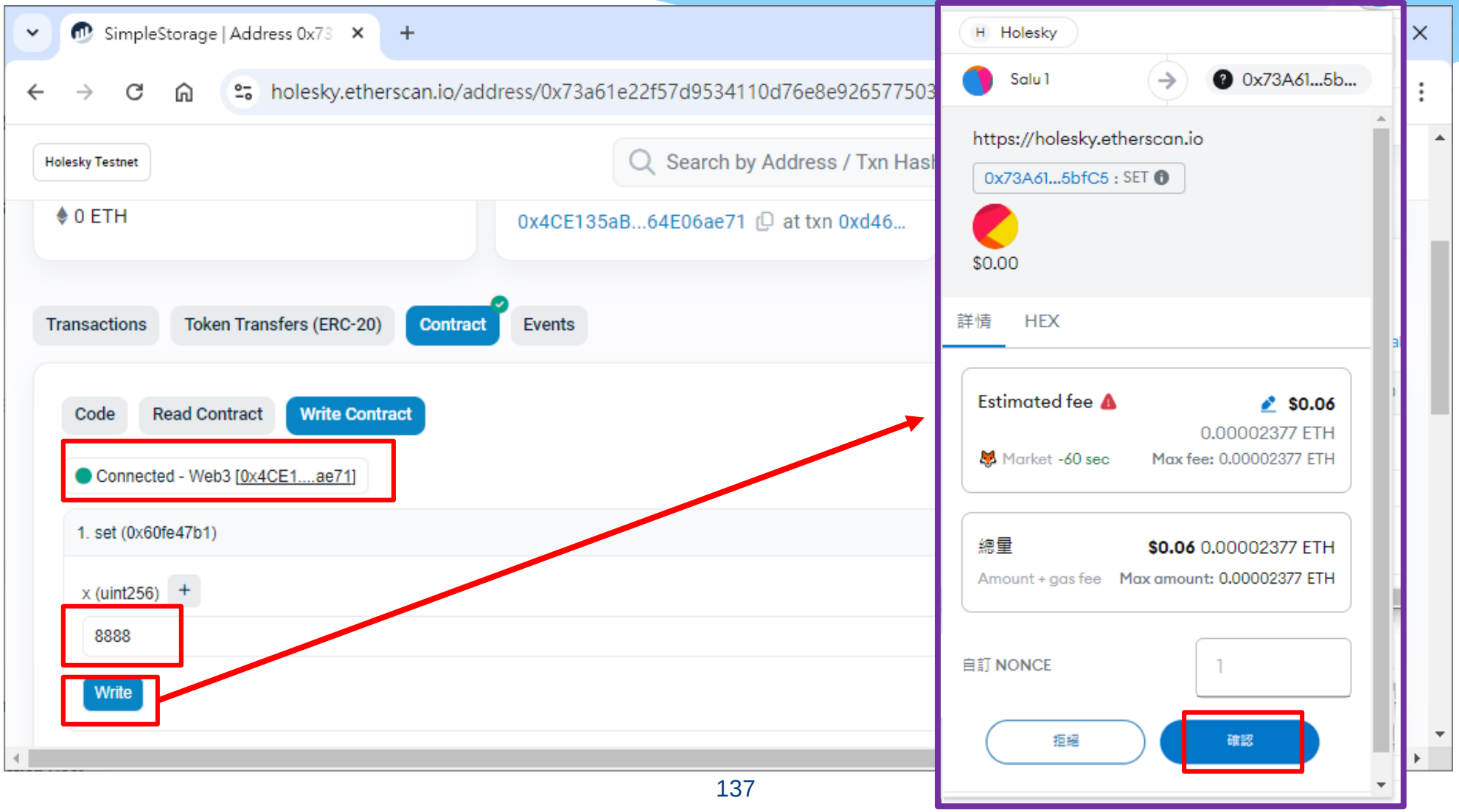


https://holesky.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5#writeContract

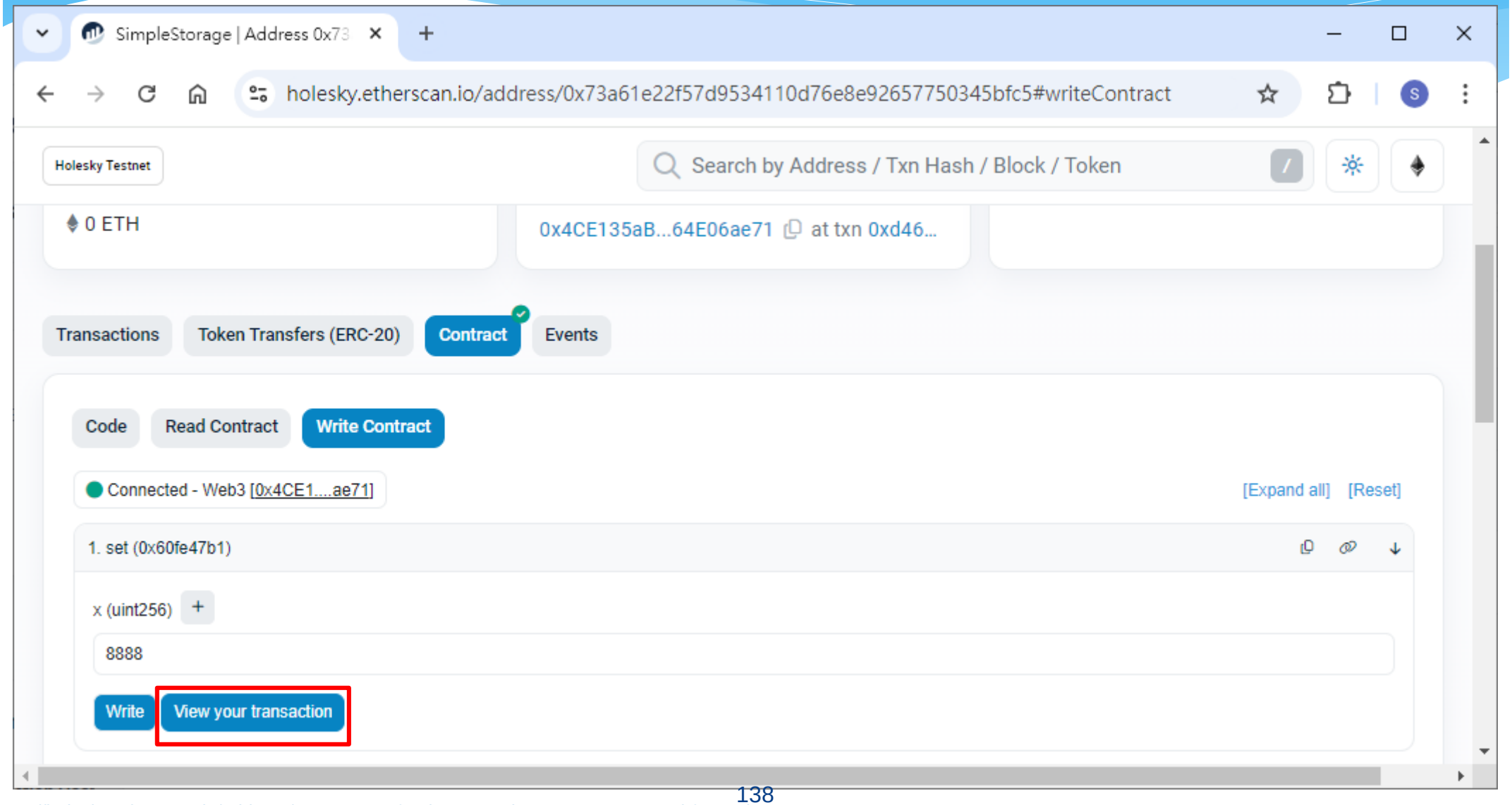
Connected-Web3



Write contract-set function

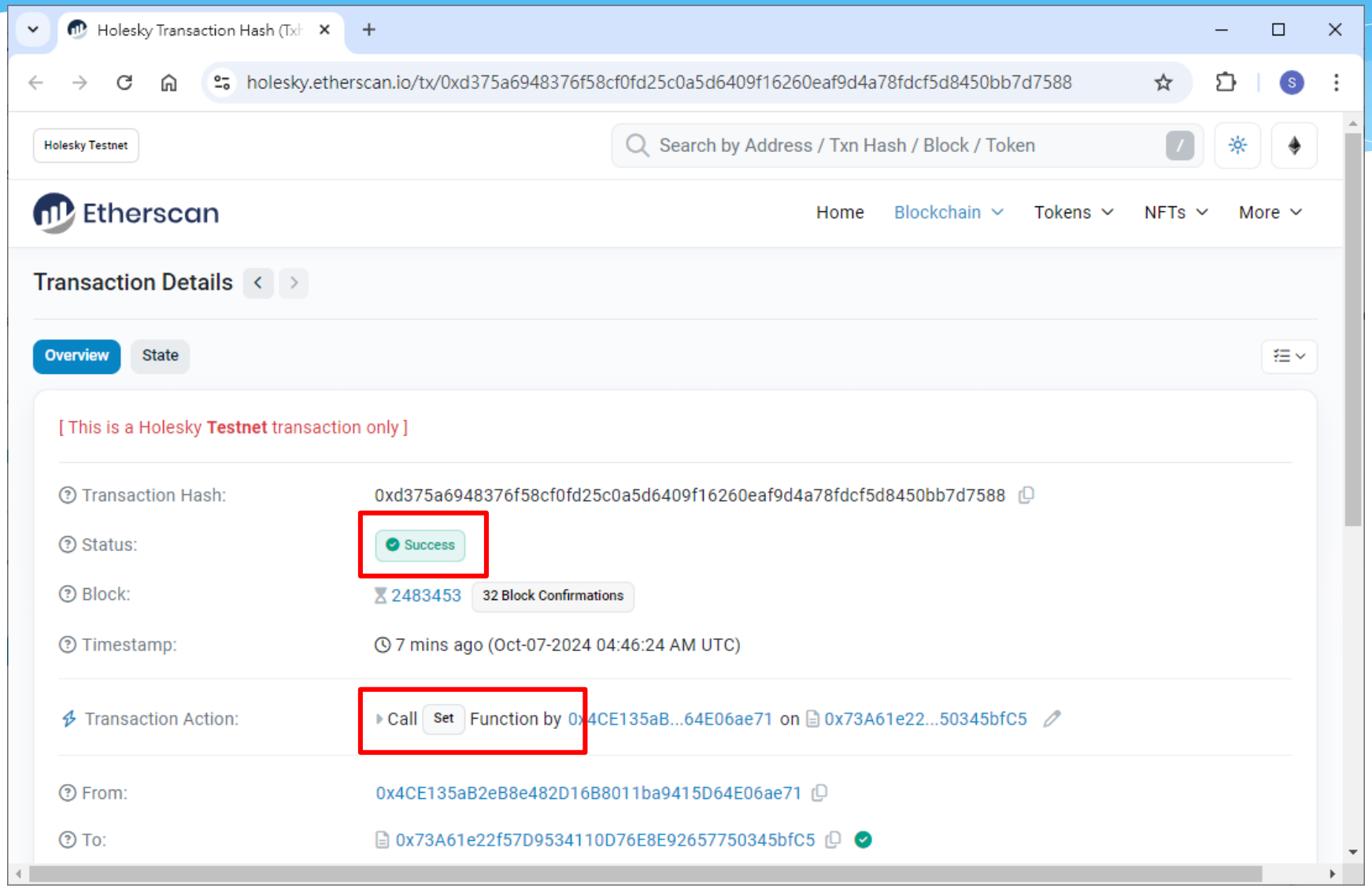


View your transaction



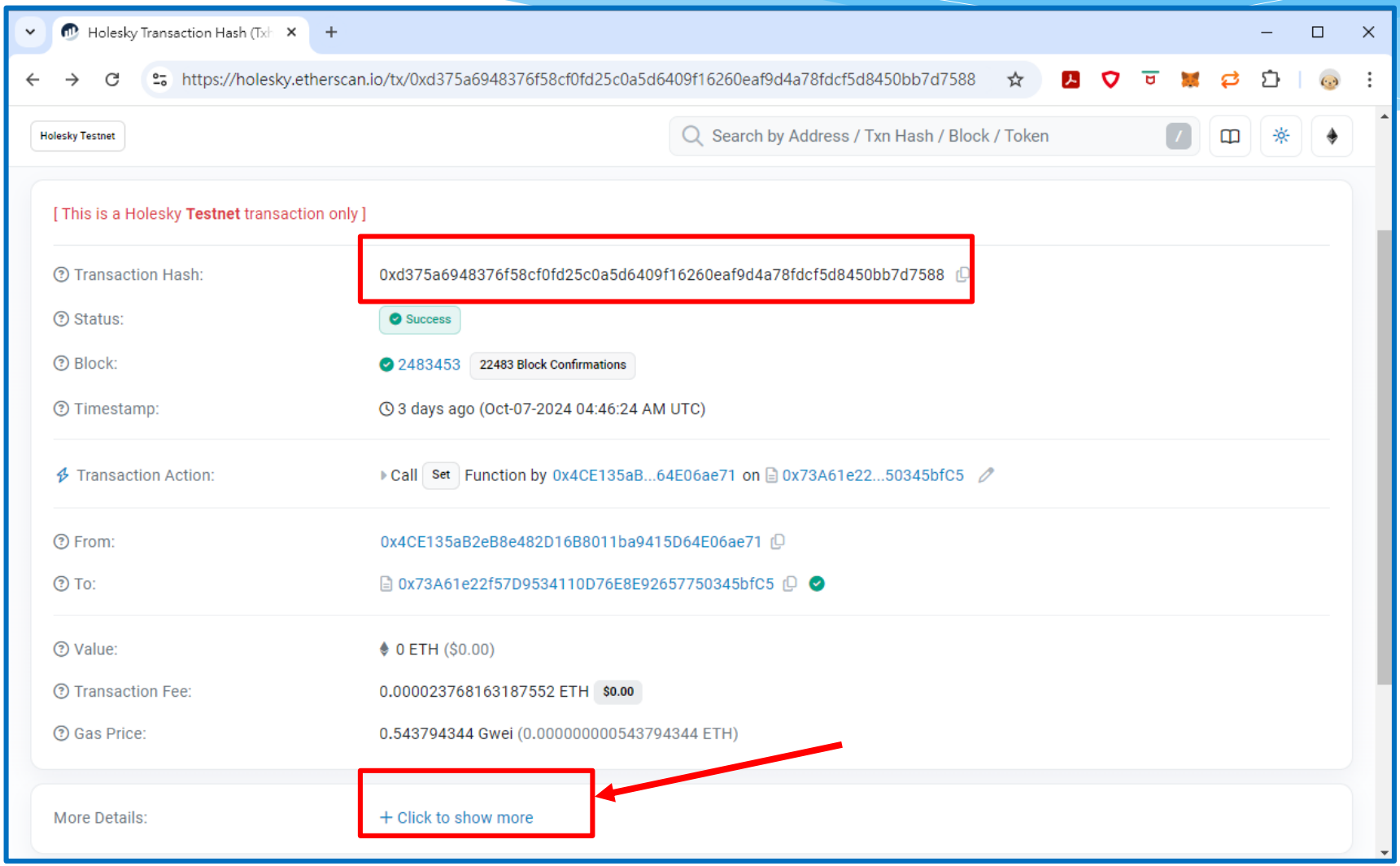
https://holesky.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5#writeContract

View your transaction



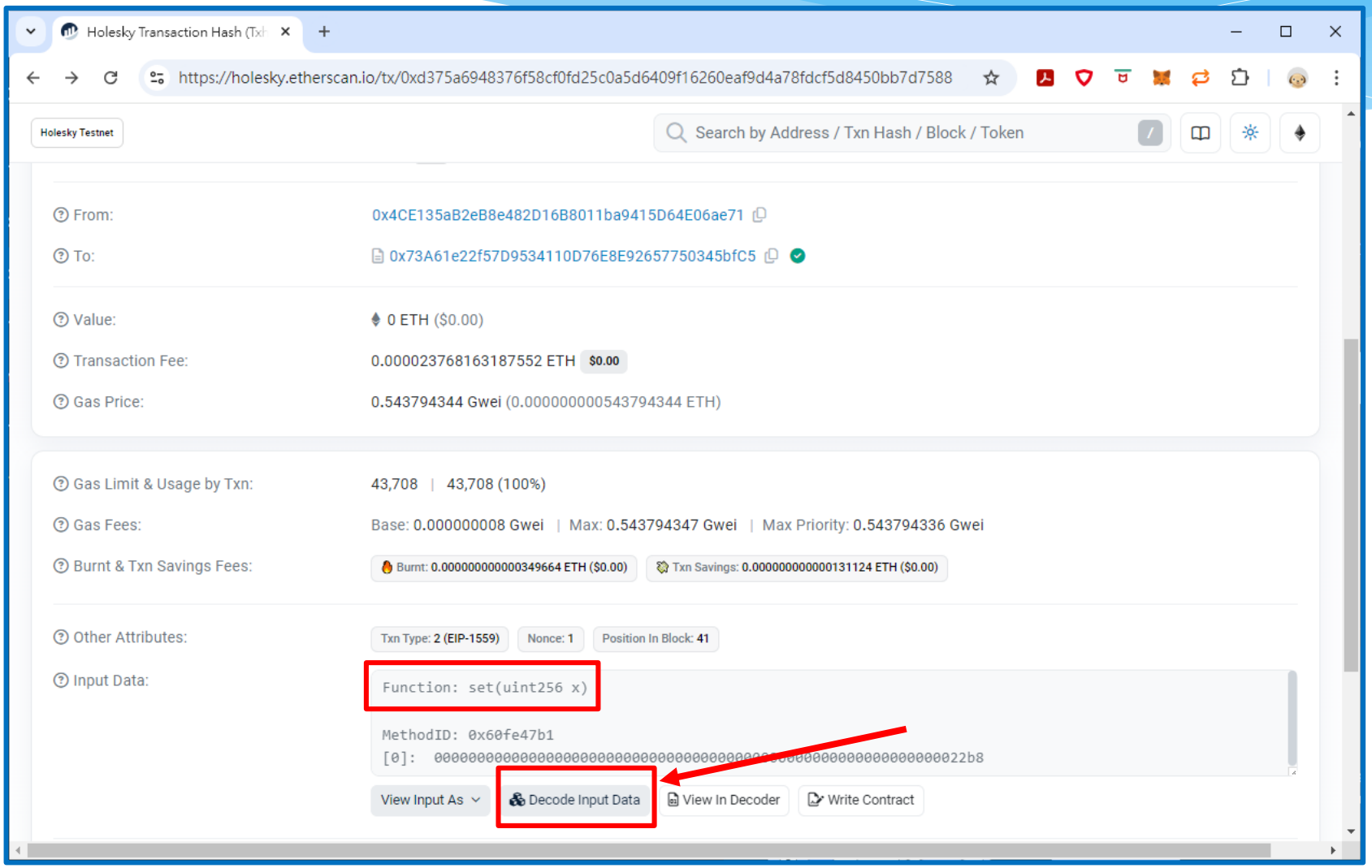
https://holesky.etherscan.io/tx/0xd375a6948376f58cf0fd25c0a5d6409f16260eaf9d4a78fdcf5d8450bb7d7588

View your transaction-More Detail



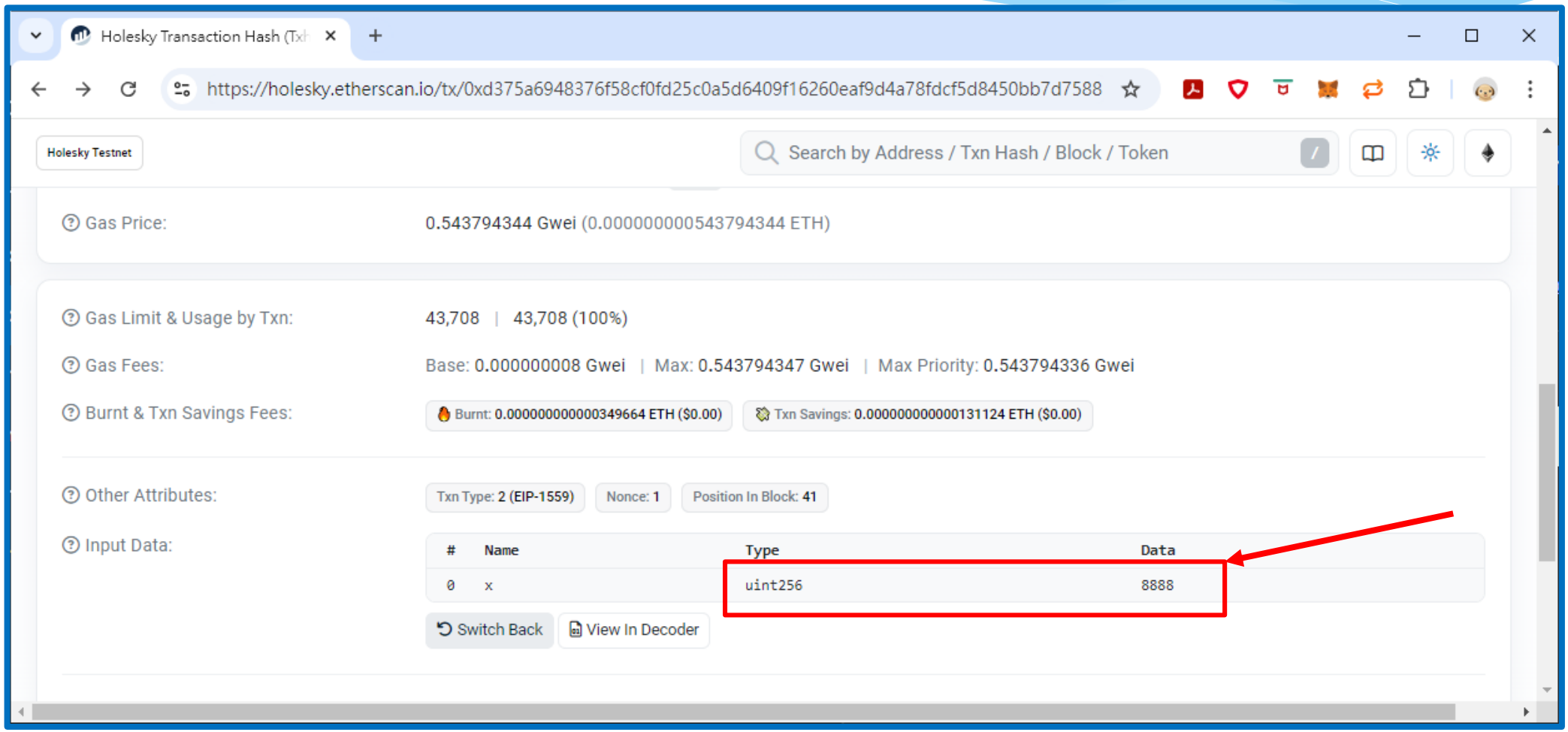
https://holesky.etherscan.io/tx/0xd375a6948376f58cf0fd25c0a5d6409f16260eaf9d4a78fdcf5d8450bb7d7588

View your transaction-Input Data



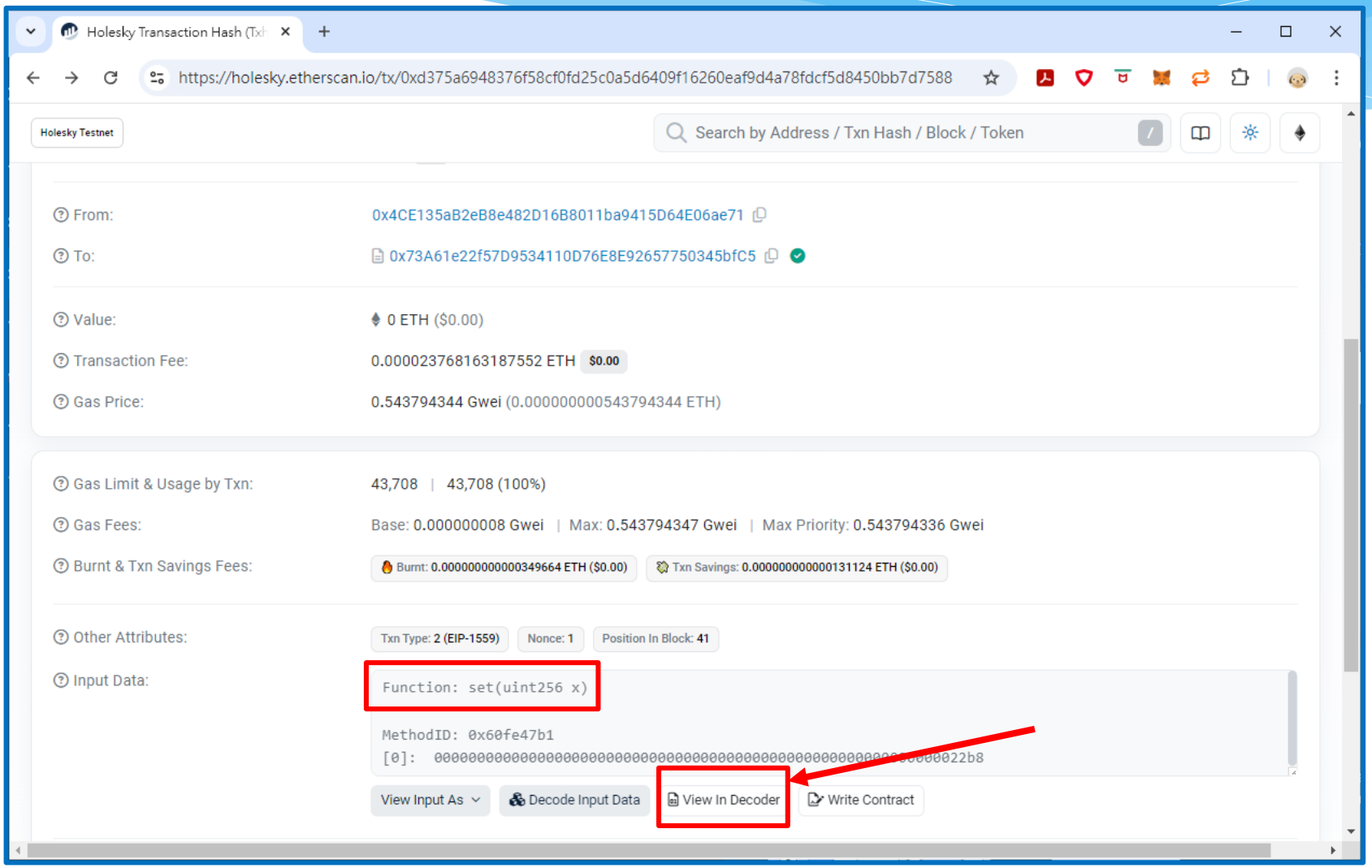
https://holesky.etherscan.io/tx/0xd375a6948376f58cf0fd25c0a5d6409f16260eaf9d4a78fdcf5d8450bb7d7588

View your transaction-Decode Input Data



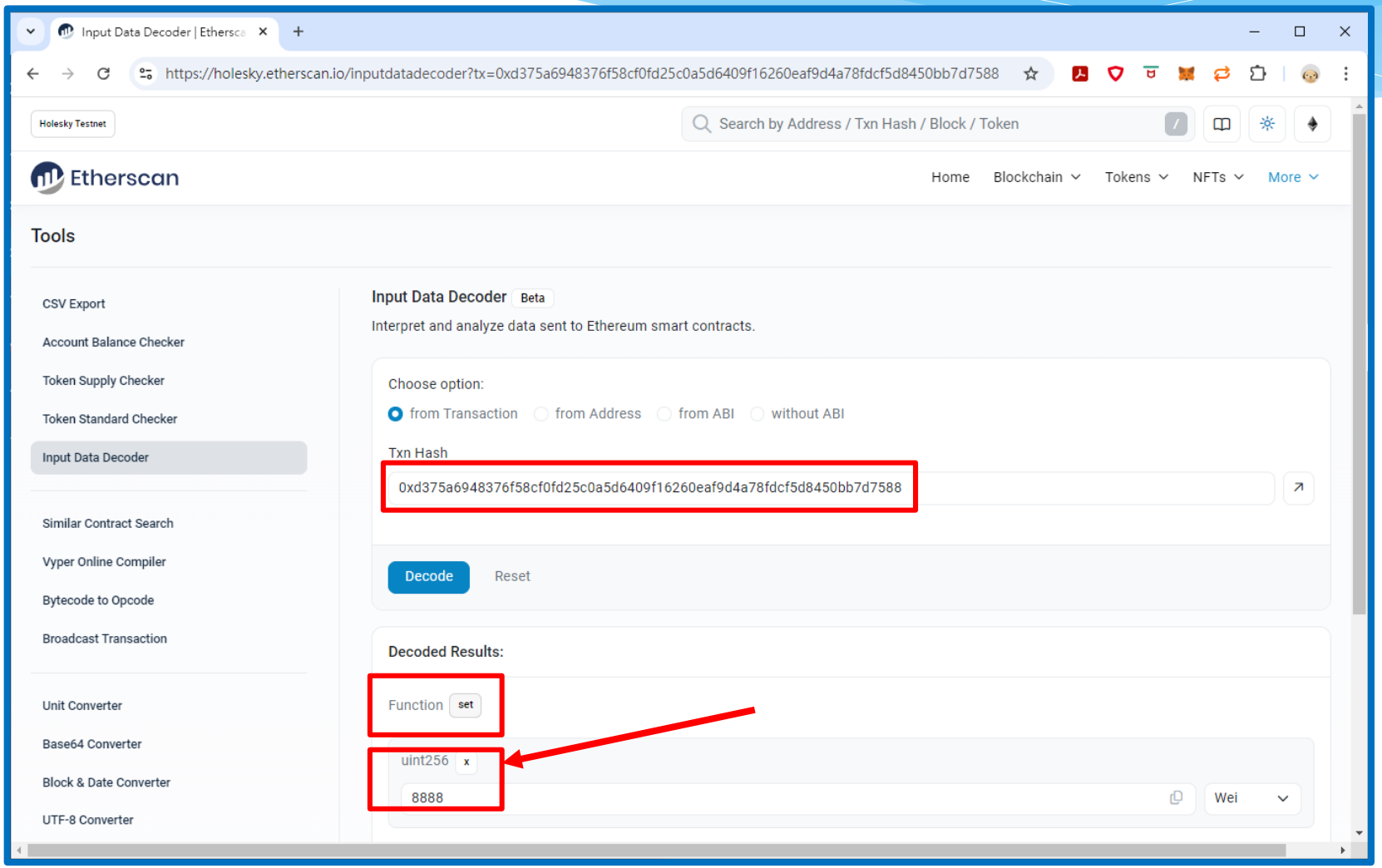
https://holesky.etherscan.io/tx/0xd375a6948376f58cf0fd25c0a5d6409f16260eaf9d4a78fdcf5d8450bb7d7588

View your transaction-Input Data



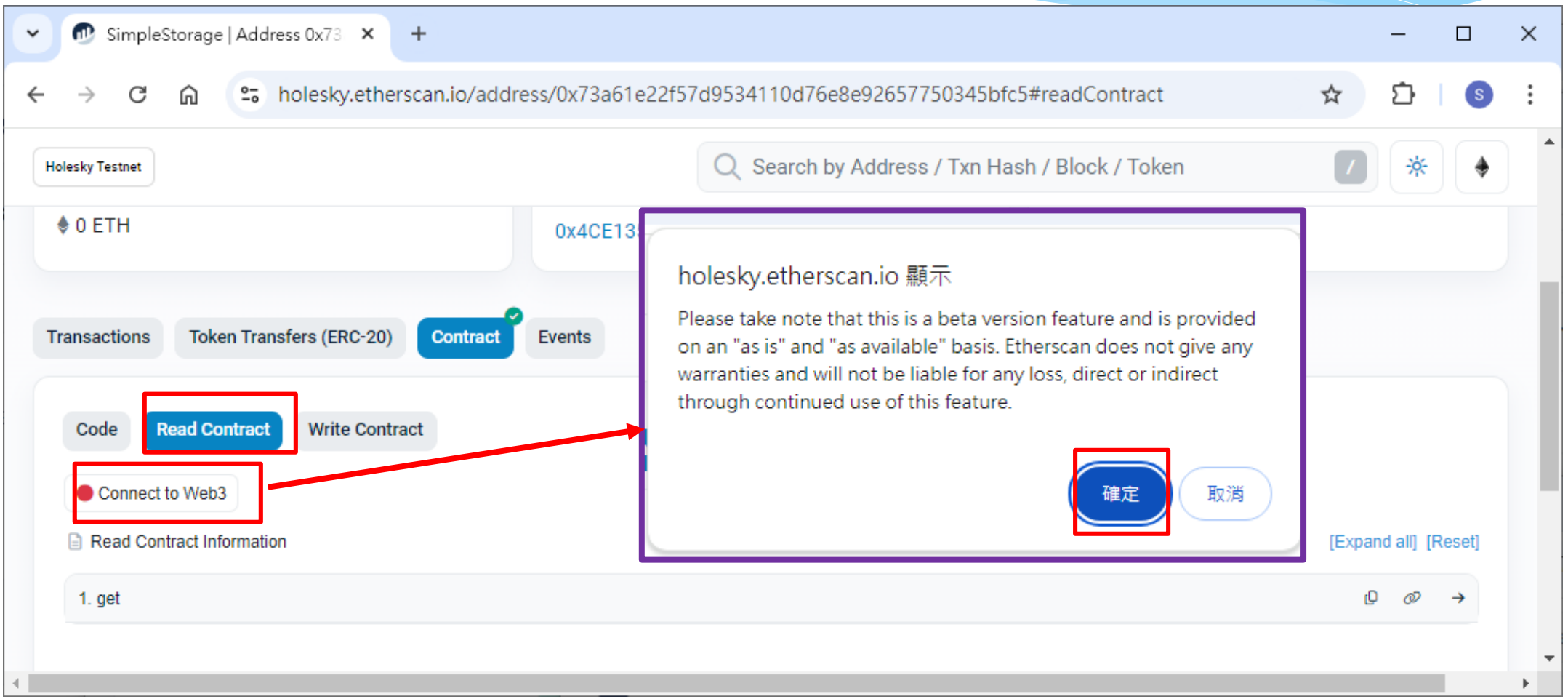
https://holesky.etherscan.io/tx/0xd375a6948376f58cf0fd25c0a5d6409f16260eaf9d4a78fdcf5d8450bb7d7588

View your transaction-Input Data Decoder



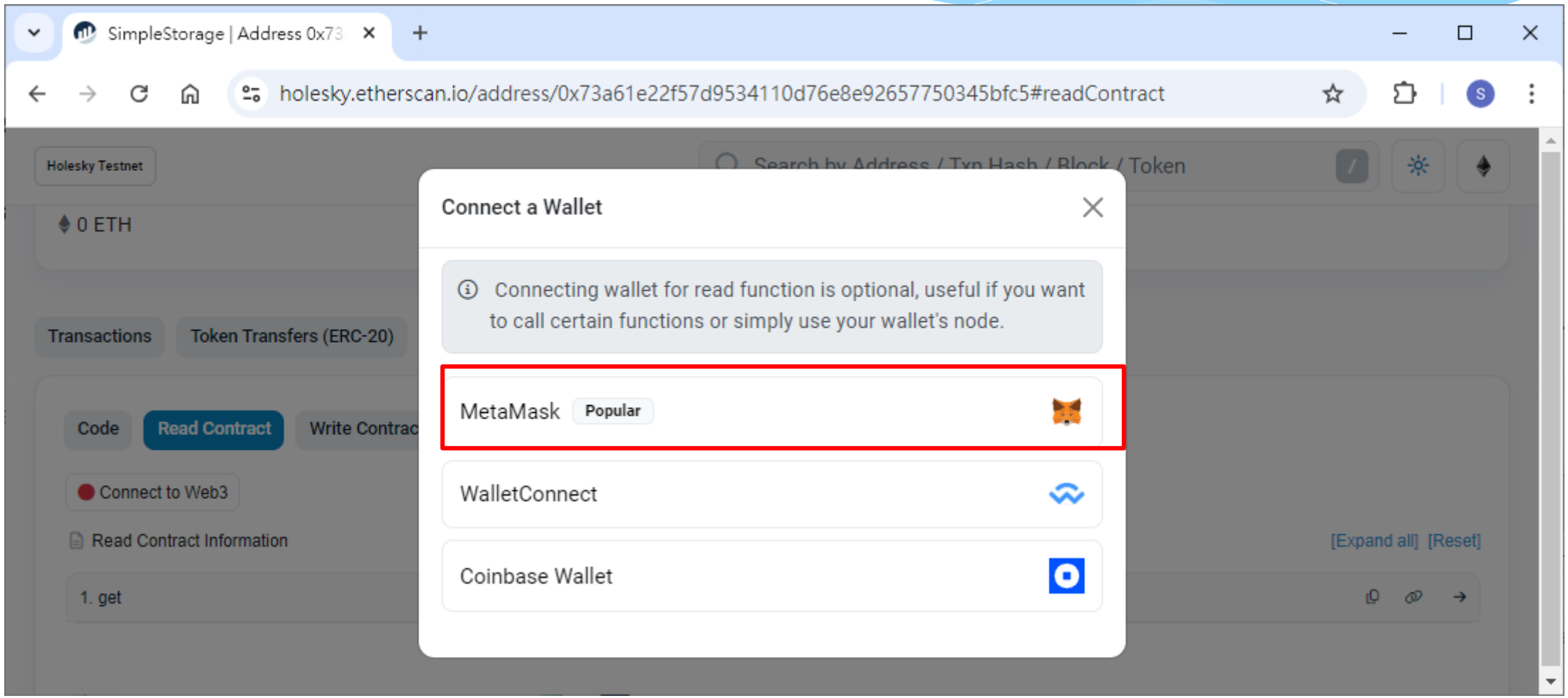
https://holesky.etherscan.io/tx/0xd375a6948376f58cf0fd25c0a5d6409f16260eaf9d4a78fdcf5d8450bb7d7588

Connect to Web3



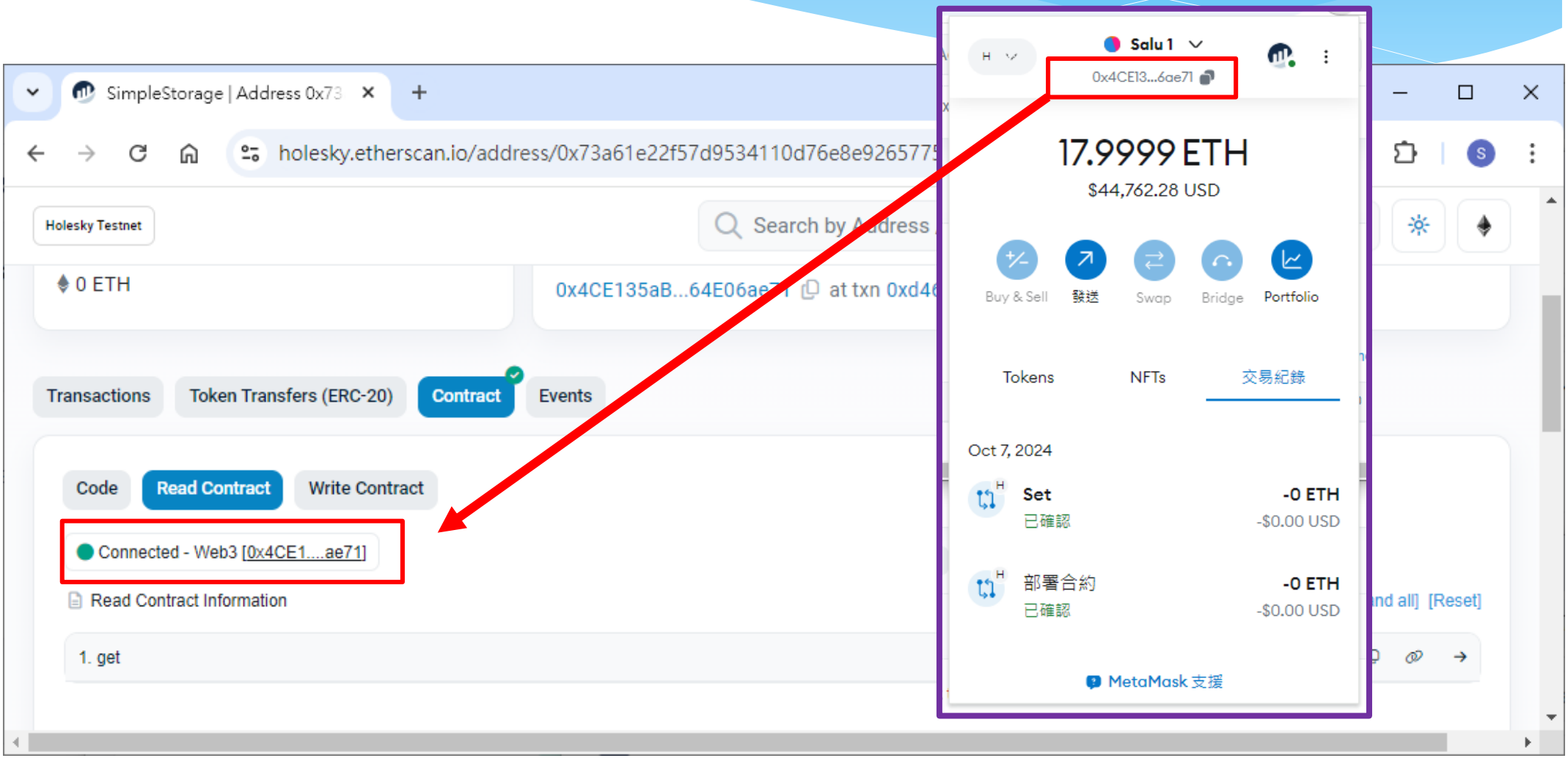
https://holesky.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5#readContract

Connect a Wallet-MetaMask



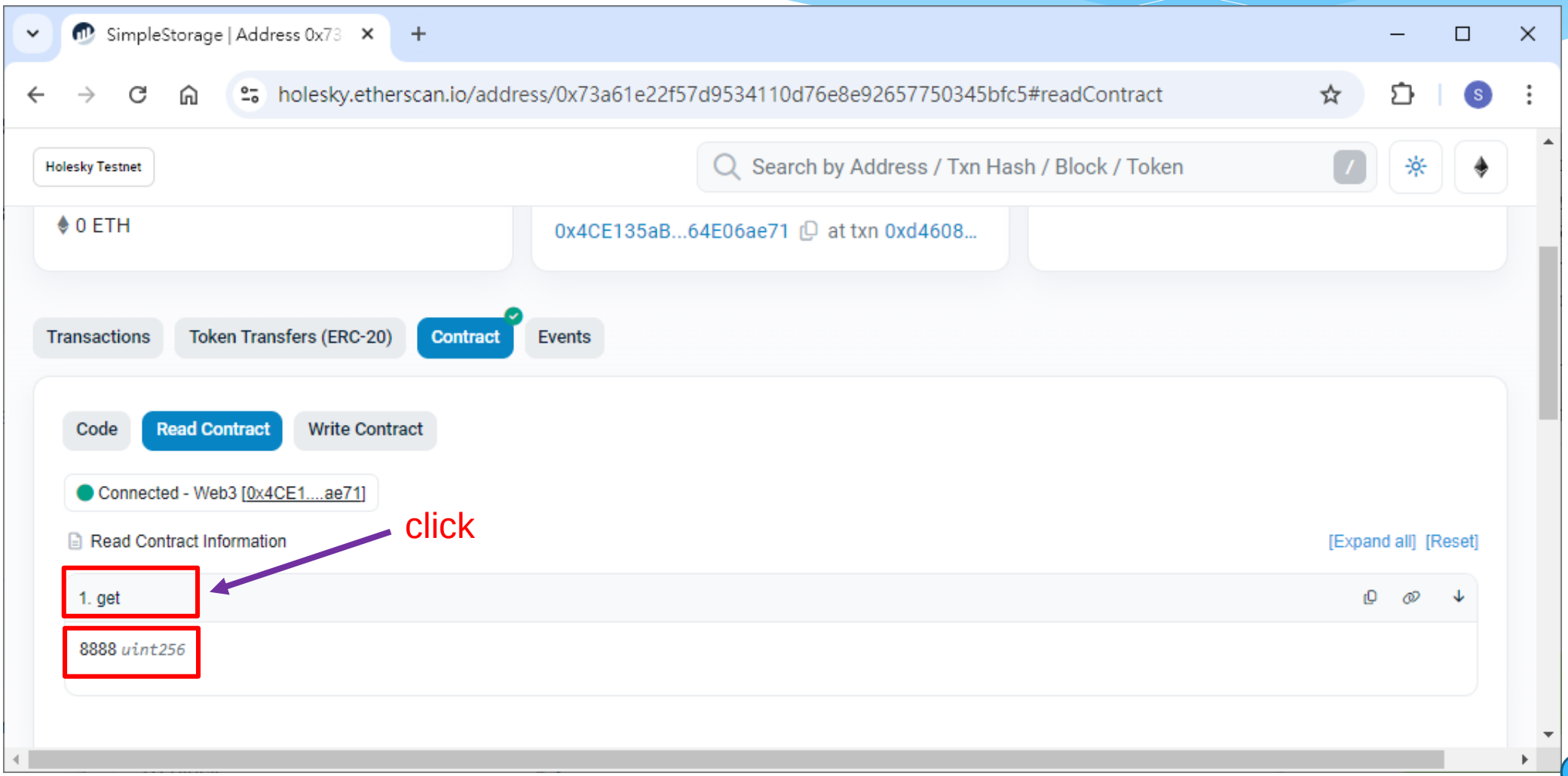
https://holesky.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5#readContract

Connected Web3 using account



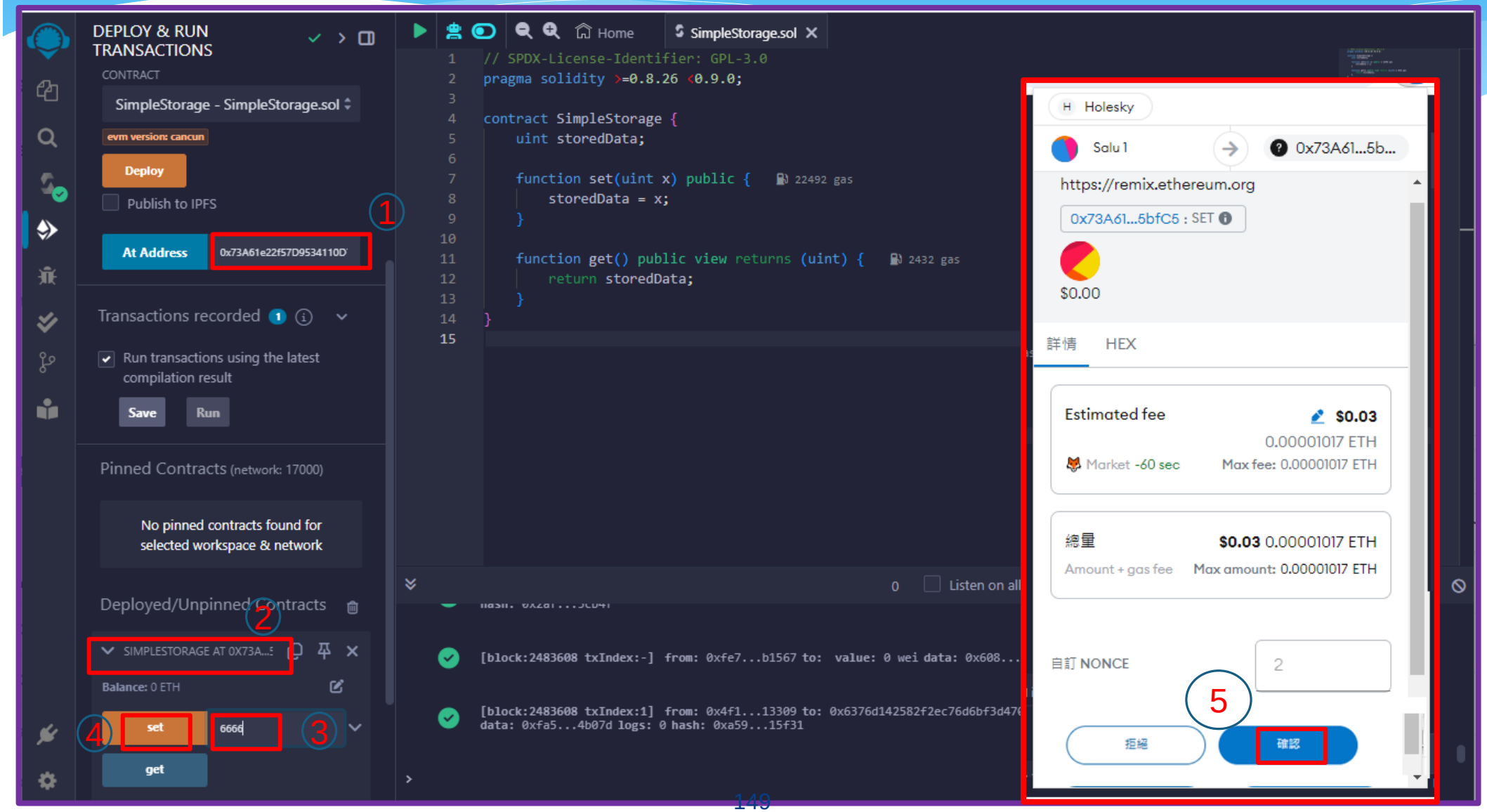
https://holesky.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5#readContract

Read Contract-get function

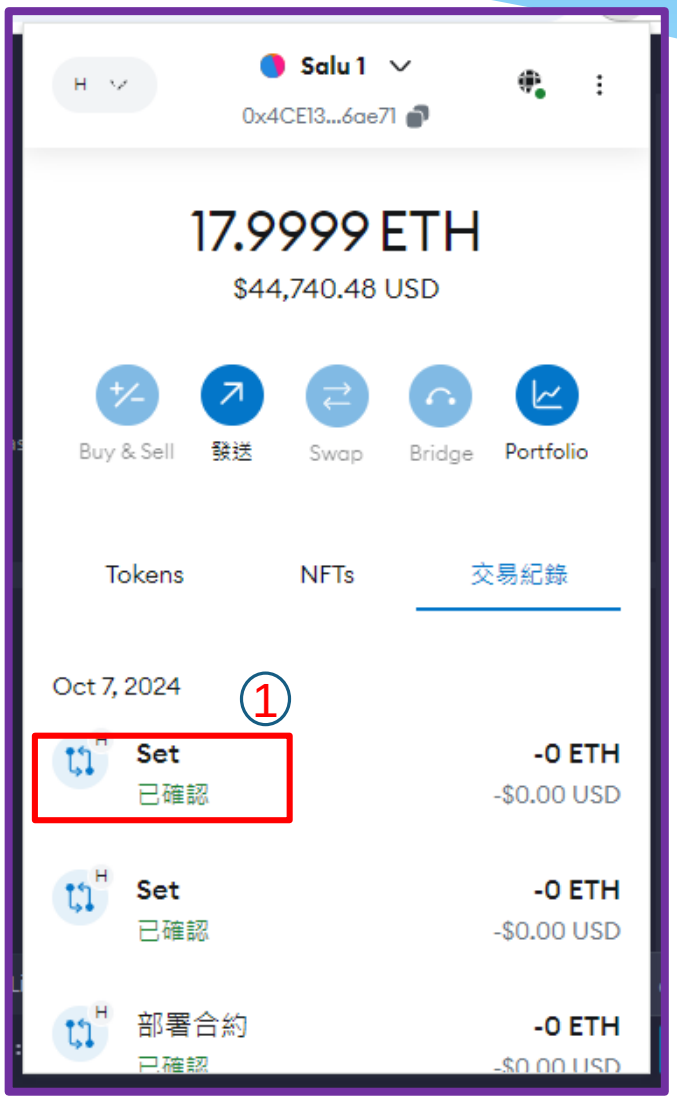


https://holesky.etherscan.io/address/0x73a61e22f57d9534110d76e8e92657750345bfc5#readContract

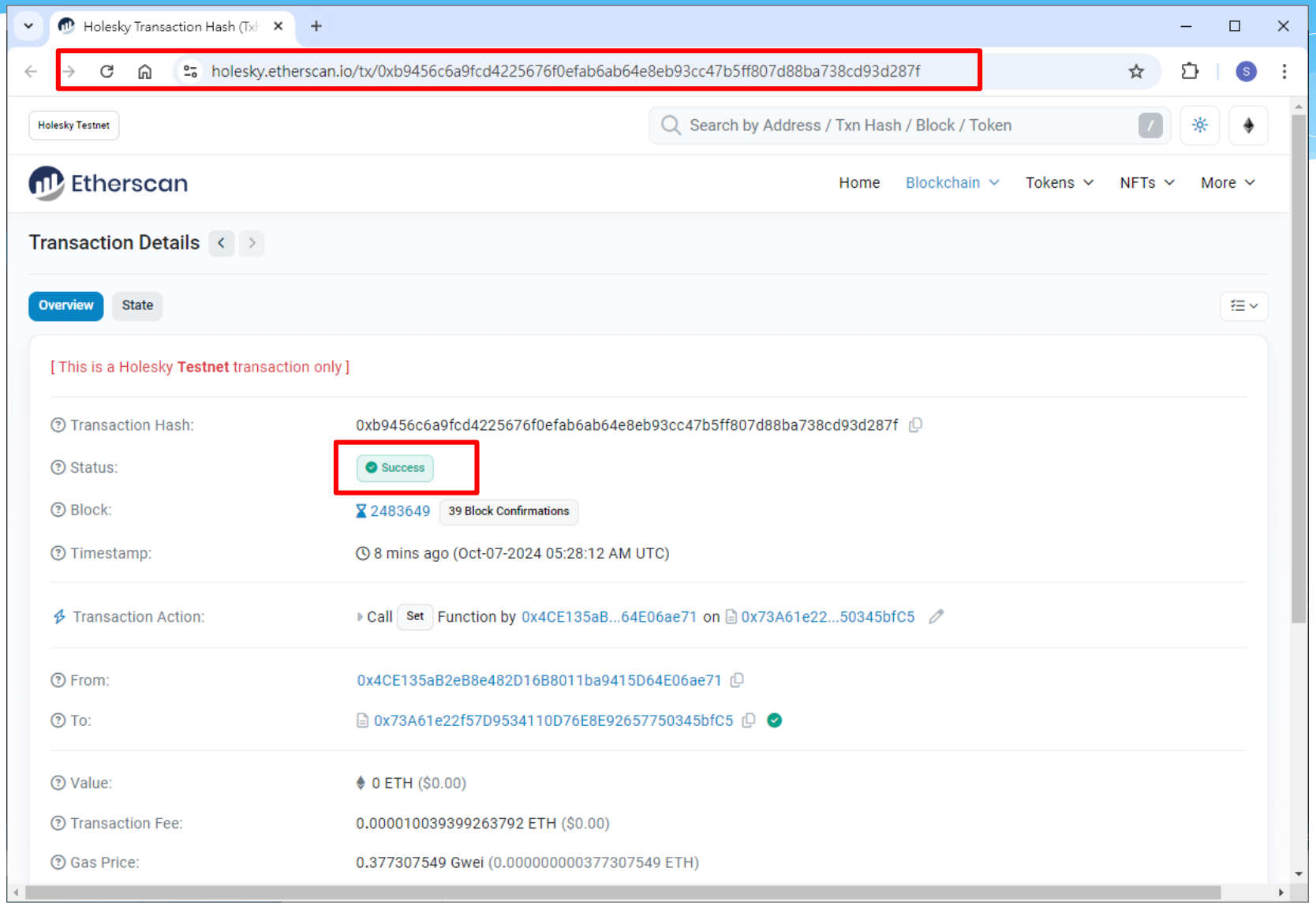
set Contract using Remix



View transaction on etherscan

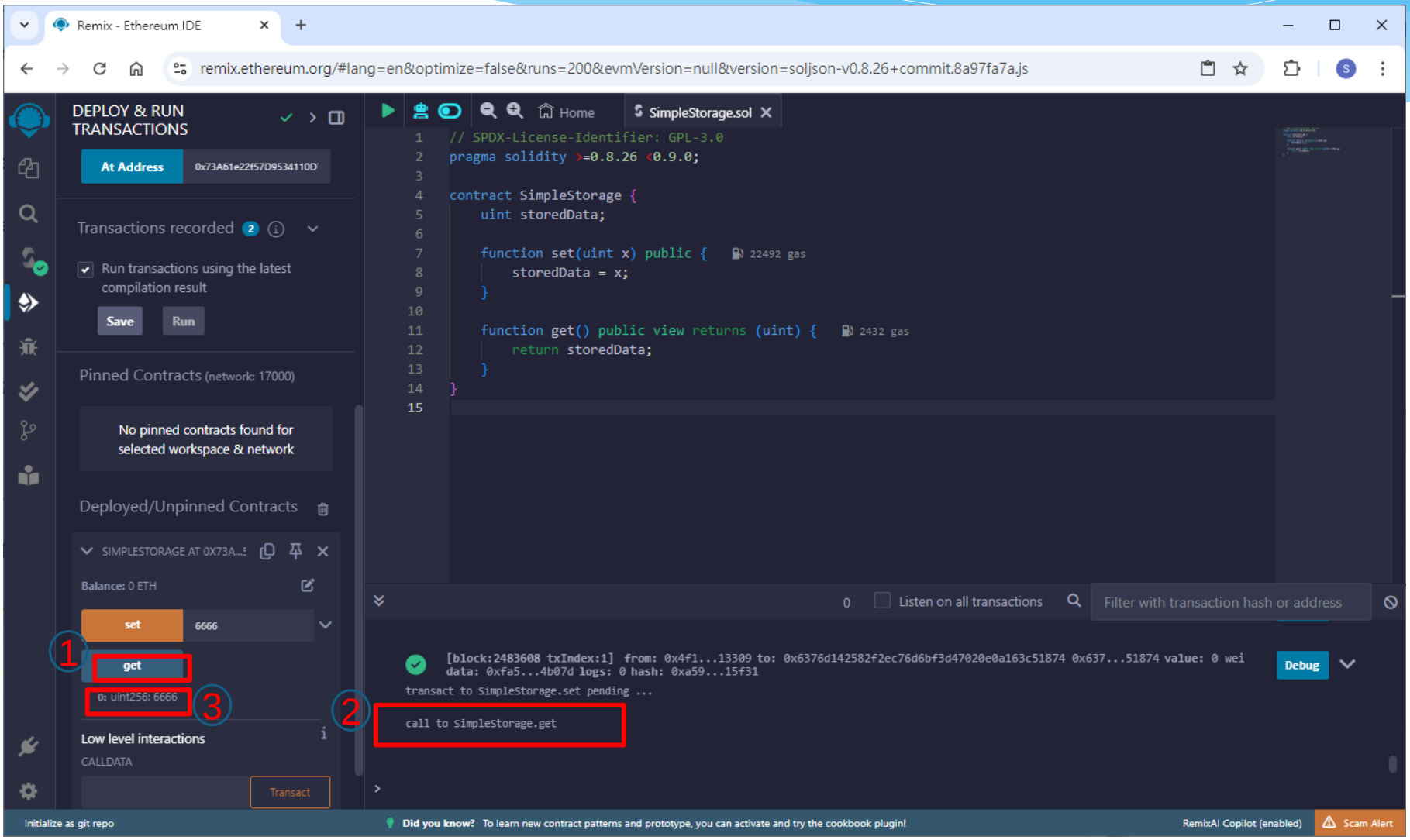


View transaction on etherscan



https://holesky.etherscan.io/tx/0xb9456c6a9fcd4225676f0efab6ab64e8eb93cc47b5ff807d88ba738cd93d287f

get Contract using Remix



References

References

- MetaMask
  https://github.com/MetaMask/metamask-extension
- Remix
  https://github.com/ethereum/browser-solidity
- Go Ethereum
  https://github.com/ethereum/go-ethereum
- Solc (Solidity compiler)
  https://github.com/ethereum/solidity
- Mist (Ethereum Wallet)
  https://github.com/ethereum/mist
- Truffle
  https://github.com/trufflesuite/truffle

References

- Ethereum EVM
  https://github.com/takenobu-hs
- Stack Exchange: Ethereum block architecture
  https://ethereum.stackexchange.com/questions/268/ethereum-block-architecture/6413
- Diving Into The Ethereum VM
  https://blog.qtum.org/diving-into-the-ethereum-vm-6e8d5d2f3c30
- Awesome Ethereum Virtual Machine
  https://github.com/pirapira/awesome-ethereum-virtual-machine

References

- Ethereum

  https://github.com/ethereum/
- Solidity Documentation

  https://solidity.readthedocs.io

  https://docs.soliditylang.org/en/v0.8.28/
- Web3 JavaScript app API for 0.2x.x

  https://github.com/ethereum/wiki/wiki/JavaScript-API
- Ledger

  https://hyperledger-fabric.readthedocs.io/en/release-2.2/ledger/ledger.html