



# Cómo crear tu propia moneda desde 0

Basado en la creación de la moneda de Ingeniero Binario



## Contenido

Introducción.....	3
¿Dónde desarrollamos? .....	4
Tu primer contrato inteligente .....	7
¿Qué es un token?.....	13
Creando un token ERC20 .....	14
Publicando tu moneda .....	24
Cómo enviar monedas .....	50
Cómo recibir monedas .....	54



# Introducción

Los contratos inteligentes permiten resolver problemas comunes de una manera que maximiza la confianza. El propósito de los contratos es reducir la ambigüedad y la inclinación para que se produzca un conjunto predecible de resultados y se pueda confiar en ellos.

En Ethereum, puedes escribir contratos inteligentes con el lenguaje de programación completo Turing integrado (Solidity). El lenguaje puede crear sus propias reglas arbitrarias para la propiedad, los formatos de transacción y las funciones de transición de estado. Los contratos inteligentes en Ethereum están escritos con Solidity y están destinados a ejecutarse en la máquina virtual Ethereum (EVM).

Para ejecutar realmente el código de contrato inteligente, alguien tiene que enviar suficiente Ether como tarifa de transacción. La tarifa se calcula en función de los recursos informáticos necesarios. Esto paga a los nodos mineros por participar y proporcionar su poder de cómputo.

Este documento cubre todos los aspectos esenciales para escribir un contrato inteligente desde cero, compilar el contrato e implementarlo en una red Ethereum.

Los contratos inteligentes que se dan en este documento están dirigidos al compilador de Solidity 0.7.6. Las nuevas versiones se lanzan rápidamente y te recomendamos que consultes el registro de cambios para Ethereum y Solidity en sus respectivos [repositorios de GitHub](#). Los contratos inteligentes en este libro son solo para fines ilustrativos. Se recomienda encarecidamente probar y auditar el código en busca de errores antes de implementarlo en la red principal.



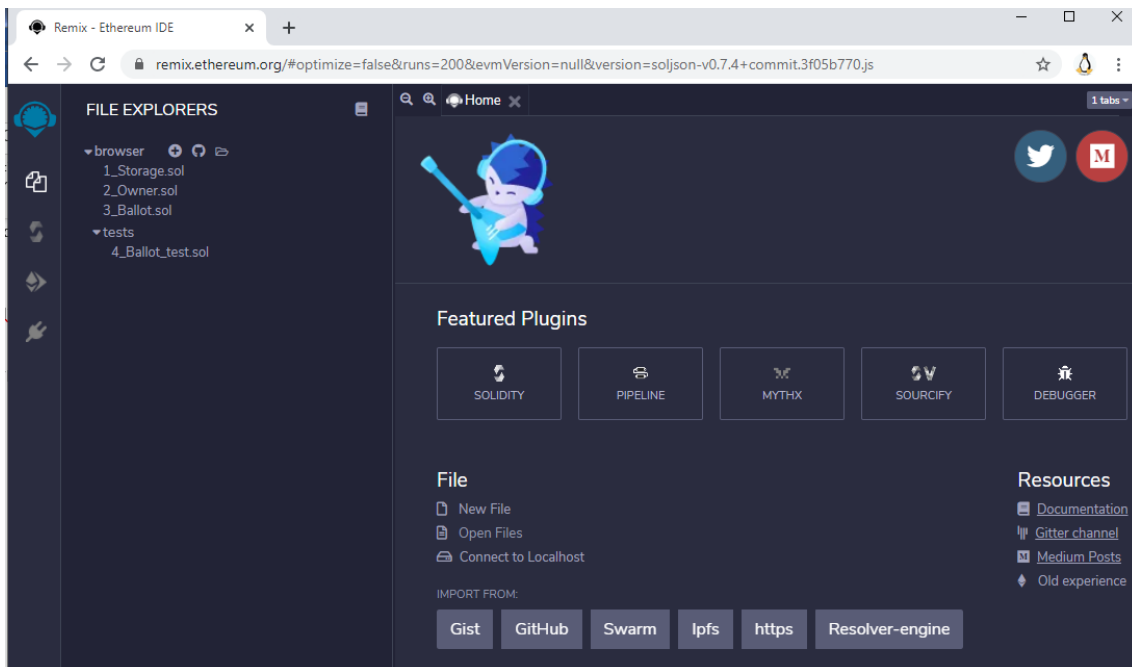
# ¿Dónde desarrollamos?

Desarrollar una aplicación distribuida a menudo requiere que el desarrollador interactúe con múltiples herramientas e idiomas. Existen entornos de desarrollo integrados (IDE) eficientes que pueden ayudarte a realizar la mayor parte del trabajo relacionado con el desarrollo de una aplicación. Los IDE suelen combinar editores de código, compiladores, depuradores y otras herramientas útiles. A algunos les puede resultar cómodo usar un editor de texto simple y herramientas de línea de comandos relacionadas para el desarrollo.

Uno de los IDE más populares usados para Solidity es Remix. Veamos cómo puedes usarlo para tu desarrollo.

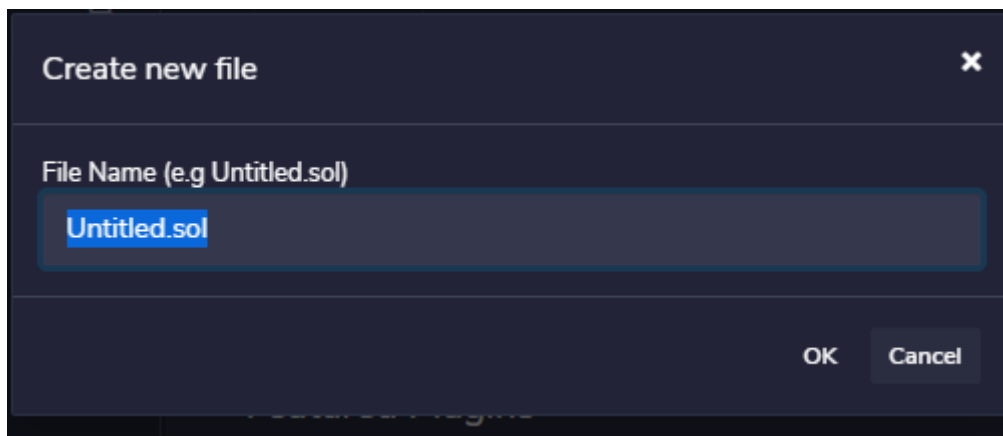
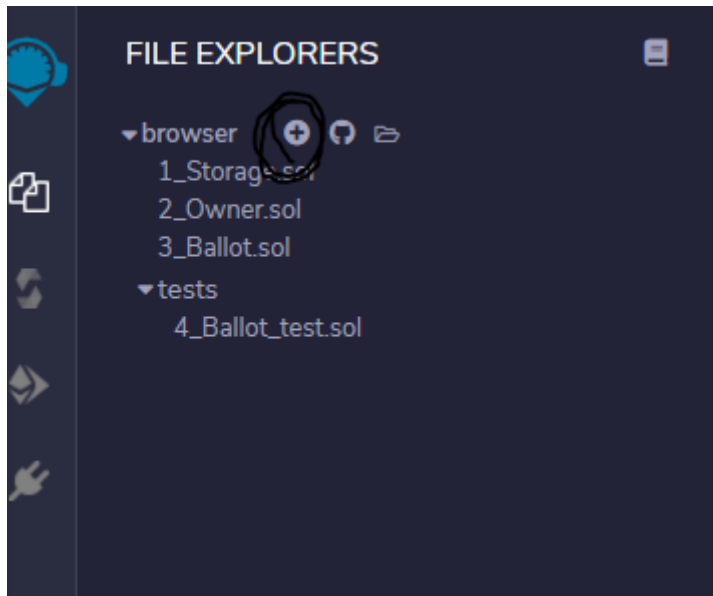
Remix es un IDE basado en web que puede utilizar para escribir, probar, depurar e implementar contratos inteligentes.

[Accede aquí](#)



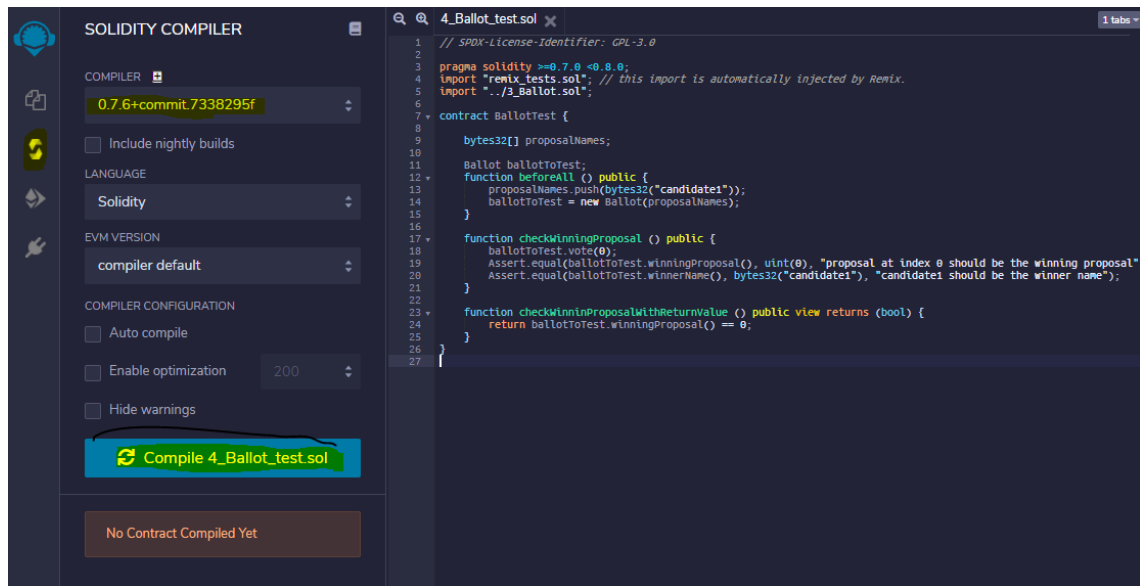


Utiliza la vista de archivos del panel izquierdo para crear y eliminar archivos.

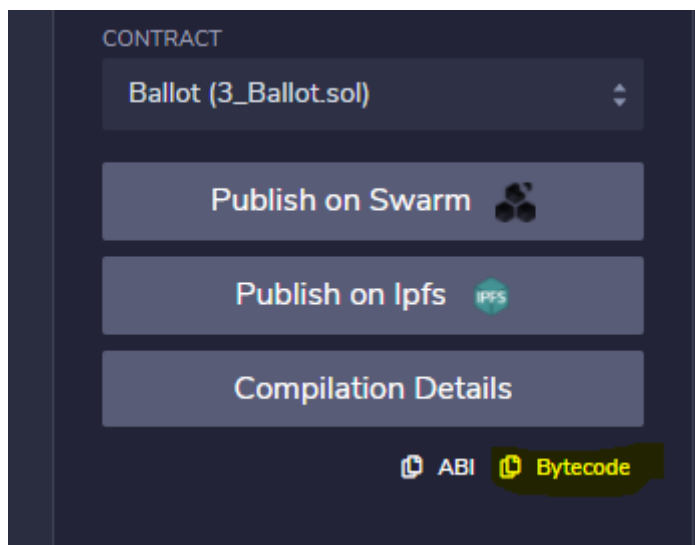


Utiliza el código de prueba y compílalo con la versión de Solidity 0.7.6.





Una vez compilado puedes observar que se puede obtener la traducción a código de bajo nivel ByteCode





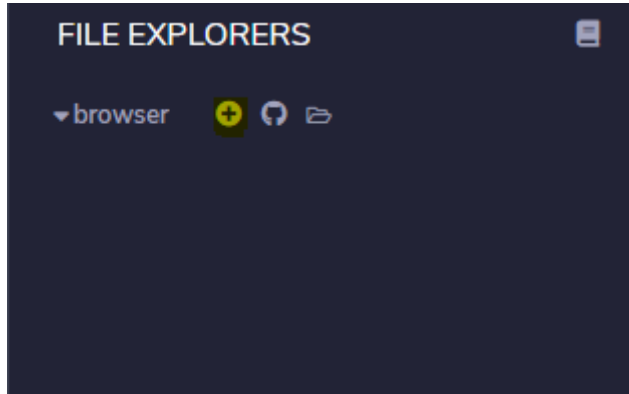
# Tu primer contrato inteligente

Solidity es el idioma preferido para escribir contratos inteligentes en Ethereum. Un contrato inteligente de Solidity es una colección de código (funciones) y datos (estado) que residen en una dirección específica en la cadena de bloques Ethereum.

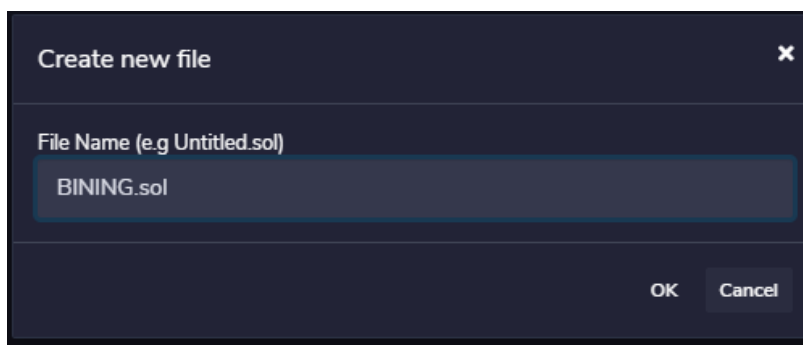
Solidity es un lenguaje de alto nivel de tipo estático que está influenciado por JavaScript, Python y C ++. Solidity admite herencias, bibliotecas y tipos definidos por el usuario, y está diseñado para EVM (Ethereum Virtual Machine).

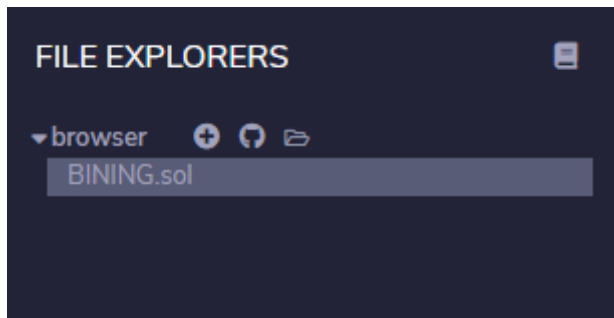
Para escribir tu contrato inteligente se utilizará el compilador Remix, que ya deberías tener preparado para ello.

Eliminar todos los archivos de prueba y crear uno con el nombre que quieras para tu moneda.



En mi caso es BINING



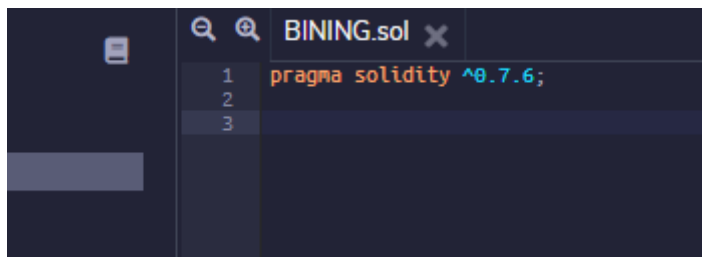


Empezarás escribiendo el famoso primer “Hola Mundo”.

Para ello, es necesario especificar la versión del compilador de destino utilizando la versión pragma.

Se recomienda especificar la versión del compilador de destino para cada contrato. Esto evitará que el contrato se compile en versiones futuras o anteriores, que pueden tener cambios incompatibles.

```
> pragma solidity ^0.4.21;
```



Este código solo se compilará con la versión 0.7.6 de Solidity

Tu primer contrato quedaría de esta forma.

```
pragma solidity ^0.7.6;

contract Bining {
    string textToPrint = "Mi primer contrato inteligente";

    function changeText(string memory _text) public {
        textToPrint = _text;
    }

    function printSomething() public view returns (string memory) {
        return textToPrint;
    }
}
```

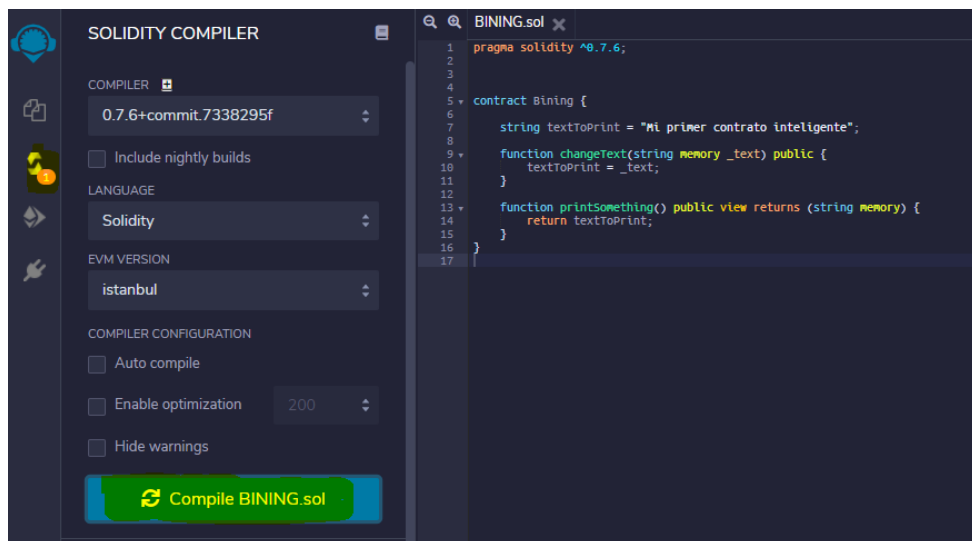




```
.runs=200&evmVersion=istanbul&version=soljson-v0.7.6+commit.7338295f.js
```

```
1  pragma solidity ^0.7.6;
2
3
4
5  contract Bining {
6
7      string textToPrint = "Mi primer contrato inteligente";
8
9      function changeText(string memory _text) public {
10         textToPrint = _text;
11     }
12
13     function printSomething() public view returns (string memory) {
14         return textToPrint;
15     }
16 }
```

Puedes compilar el código.



Se desplegará las opciones del contrato una vez compiles. Puedes ver la información del contrato si lo deseas.





## ¿Cómo funciona?

EVM es el entorno de ejecución de los contratos inteligentes de Ethereum. Los contratos que se ejecutan dentro de EVM no tienen acceso a la red, al sistema de archivos ni a otros procesos. Esto es debido a la naturaleza de aislamiento de EVM.

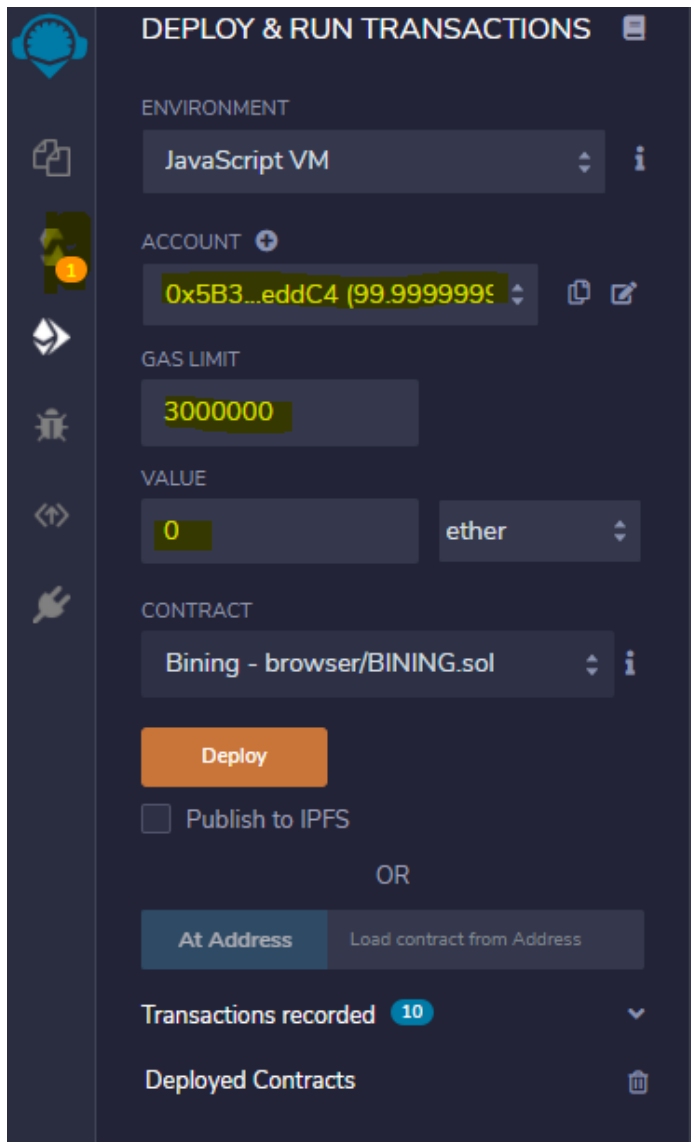
La primera línea de este contrato inteligente nos dice que el código fuente está escrito para la versión del compilador 0.7.6 o cualquier versión más reciente (como 0.8) que no interrumpa ninguna funcionalidad. La palabra clave pragma se utiliza porque los pragmas son instrucciones para el compilador sobre cómo tratar el código fuente.

La línea `string textToPrint = "hello world";` declara una variable de estado llamada `textToPrint` y le asigna un valor. Puedes pensar en ello como una ranura en una base de datos que se puede consultar y alterar llamando a las funciones en el código que administra la base de datos. Para acceder a la variable dentro del contrato, no necesitas usar esto como en otros idiomas.

Este es un contrato simple que permite a cualquier persona almacenar una cadena a la que puede acceder cualquier persona en el mundo, sin una forma de evitar que publique este valor.



Despliega el contrato inteligente simple que has creado en la red de pruebas de ETH.



Tienes varias cuentas para seleccionar, elige la que quieras. Todo es gratuito porque es un entorno de pruebas.

Una transacción puede incluir tanto datos binarios como Ether. Si la cuenta de destino contiene código, ese código se ejecuta y la carga útil (datos binarios) se proporcionan como datos de entrada. Si la cuenta de destino no existe, la transacción crea un nuevo contrato. Se ejecuta la carga útil de dicha transacción de creación de contrato. El resultado de esta ejecución se almacena permanentemente como el código del contrato.



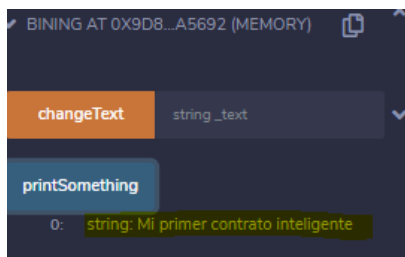
Cada transacción se carga con una cierta cantidad de gas, lo que limita la cantidad de código a ejecutar y cuánto pagar al minero por su ejecución. El precio del gas es un valor establecido por el remitente de la transacción. El remitente tiene que pagar un precio inicial, que se calcula por **gasPrice \* gas**.

Durante la ejecución, si el gas proporcionado no es suficiente, se activa una excepción y revertirá cualquier cambio de estado realizado durante la llamada. Si queda algo de gas después de la ejecución, se reembolsará a la cuenta del remitente. Todos los valores están representados en Wei, que es la denominación más pequeña de Ether (  $10^{18}$  Wei = 1 Ether ).

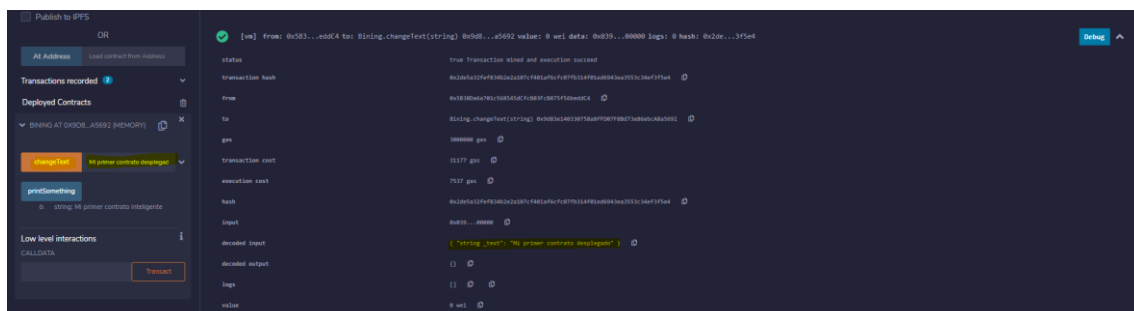
```
[vm] from: 0x5B3...eddC4 to: Bining.(constructor) value: 0 wei data: 0x608...60833 logs: 0 hash: 0xa23...9d4fd

status true Transaction mined and execution succeed
transaction hash 0xa233263528a5a8b4afc49d311f5fc9fe493fee5f2bbe4d3415c9734be59d4fd
contract address 0x0d83e140330758a8ffD07f8Bd73e86ebcA8a5692
from 0x5B38D6a701c568545dcfc083fc8875f50bedc4
to Bining.(constructor)
gas 3000000 gas
transaction cost 300056 gas
execution cost 179404 gas
hash 0xa233263528a5a8b4afc49d311f5fc9fe493fee5f2bbe4d3415c9734be59d4fd
input 0x608...60833
decoded input {}
decoded output -
logs []
value 0 wei
```

El despliegue del contrato nos permitirá cambiar el mensaje.



Escribes el texto nuevo y pulsas “changeText” y modificará el mensaje que devuelve el contrato.





Una vez tienes claro cómo crear tu primer contrato y desplegarlo en EVM (máquina virtual de Ethereum), entorno de pruebas, ya puedes crear la moneda personalizada.

## ¿Qué es un token?

Los tokens en Ethereum representan un valor financiero o existen como un activo digital. Estos tokens pueden ser fungibles o no fungibles, según el requisito. Pueden representar cualquier cosa, desde una moneda hasta un gato virtual que se puede comerciar. El uso de un token basado en Ethereum te permite hacer uso de la infraestructura existente de Ethereum en lugar de construir una cadena de bloques desde cero.

Un token fungible es aquel que no es único y es perfectamente intercambiable con otros tokens idénticos. Por ejemplo, el dólar estadounidense se puede comparar con tokens fungibles. Es perfectamente intercambiable con cualquier otro dólar estadounidense. **Los tokens no fungibles (NFT)** son únicos en la naturaleza y se pueden distinguir entre sí. Por ejemplo, una tarjeta coleccionable se puede considerar como una NFT, y cada tarjeta tiene diferentes características que la hacen destacar.

La comunidad Ethereum tiene algunos estándares definidos para varios tipos de tokens. Estos estándares se pueden utilizar para crear una variedad de uso de tokens basados en casos tales como la costumbre criptomonedas o activos. Estas monedas, fichas o activos se distribuyen a menudo al público a través de un proceso llamado **ICO (Initial Coin Offering)**.

### Nota

El termino ICO representa el proceso de distribución de activos / monedas / tokens utilizando Ethereum.



# Creando un token ERC20

Uno de los estándares de tokens más conocidos y utilizados dentro de la comunidad Ethereum es ERC20.

ERC20 hace que los activos sean fácilmente intercambiables y garantiza que puedan trabajar con las diversas aplicaciones distribuidas que siguen el mismo estándar.

En este punto vas a aprender a crear un token ERC20 y conocerás varias de sus propiedades.

Para crear un token basado en ERC20 en Ethereum, debes seguir un cierto estándar.

El contrato debe incluir las siguientes funciones:

- `totalSupply()`
- `balanceOf(address _owner)`
- `transfer(address _to, uint256 _value)`
- `transferFrom(address _from, address _to, uint256 _value)`
- `approve(address _spender, uint256 _value)`
- `allowance(address _owner, address _spender)`

El contrato debe incluir los siguientes eventos:

- `transfer(address indexed _from, address indexed _to, uint256 _value)`
- `approval(address indexed _owner, address indexed _spender, uint256 _value)`

Se añaden funciones extra también, como Burn, Mint, increaseApproval, decreaseApproval, etc. Puedes añadir más funciones para añadir funcionalidades a tu moneda.



Modifica el archivo BINING añadiendo esta porción de código.

```
pragma solidity ^0.7.6;

interface IBERC20 {

    function totalSupply() external view returns (uint256);
    function balanceOf(address account) external view returns (uint256);
    function allowance(address owner, address spender) external view
returns (uint256);

    function transfer(address recipient, uint256 amount) external returns
(bool);
    function approve(address spender, uint256 amount) external returns
(bool);
    function transferFrom(address sender, address recipient, uint256
amount) external returns (bool);
    function increaseApproval(address _spender, uint _addedValue)
external returns (bool);
    function decreaseApproval(address _spender, uint _subtractedValue)
external returns (bool);

    event Transfer(address indexed from, address indexed to, uint256
value);
    event Approval(address indexed owner, address indexed spender,
uint256 value);
    event Burn(address indexed burner, uint256 value);
    event Mint(address indexed to, uint256 amount);
    event MintFinished();
}

contract ERC20_INGENIEROBINARIO is IBERC20 {

    string public constant name = "BINING"; // --> NOMBRE DE LA MONEDA
    string public constant symbol = "BNG"; // --> ASIGNAR UN ALIAS A
LA MONEDA
    uint8 public constant decimals = 2; // --> DECIMALES QUE TENDRA
LA MONEDA
    uint256 totalSupply_ = 10000000; // --> EL TOTAL DE MONEDAS
EN CIRCULACIÓN. Si has puesto 2 decimales, el punto se desplazará dos
posiciones, quedando un total de 100000.00 monedas.(100.000)

    event Bought(uint256 amount);
    event Sold(uint256 amount);
    mapping(address => uint256) balances;

    mapping(address => mapping (address => uint256)) allowed;
```



```

    using SafeMath for uint256;

    constructor() public {
        balances[msg.sender] = totalSupply_;
    }

    function totalSupply() public override view returns (uint256) {
        return totalSupply_;
    }

    function balanceOf(address tokenOwner) public override view returns
(uint256) {
        return balances[tokenOwner];
    }

    function transfer(address receiver, uint256 numTokens) public
override returns (bool) {
        require(numTokens <= balances[msg.sender]);
        balances[msg.sender] = balances[msg.sender].sub(numTokens);
        balances[receiver] = balances[receiver].add(numTokens);
        emit Transfer(msg.sender, receiver, numTokens);
        return true;
    }

    function approve(address delegate, uint256 numTokens) public override
returns (bool) {
        allowed[msg.sender][delegate] = numTokens;
        emit Approval(msg.sender, delegate, numTokens);
        return true;
    }

    function allowance(address owner, address delegate) public override
view returns (uint) {
        return allowed[owner][delegate];
    }

    function transferFrom(address owner, address buyer, uint256
numTokens) public override returns (bool) {
        require(numTokens <= balances[owner]);
        require(numTokens <= allowed[owner][msg.sender]);

        balances[owner] = balances[owner].sub(numTokens);
        allowed[owner][msg.sender] =
allowed[owner][msg.sender].sub(numTokens);
        balances[buyer] = balances[buyer].add(numTokens);
        emit Transfer(owner, buyer, numTokens);
        return true;
    }

    function increaseApproval(address _spender, uint _addedValue) public
override returns (bool) {
        allowed[msg.sender][_spender] =
(allowed[msg.sender][_spender].add(_addedValue));
        emit Approval(msg.sender, _spender,
allowed[msg.sender][_spender]);
        return true;
    }

```





```

function decreaseApproval(address _spender, uint _subtractedValue)
public override returns (bool) {

    uint oldValue = allowed[msg.sender][_spender];

    if (_subtractedValue > oldValue) {
        allowed[msg.sender][_spender] = 0;
    } else {
        allowed[msg.sender][_spender] =
oldValue.sub(_subtractedValue);
    }

    emit Approval(msg.sender, _spender,
allowed[msg.sender][_spender]);
    return true;
}

IBERC20 public token;

function buy() payable public {
    uint256 amountTobuy = msg.value;
    uint256 dexBalance = token.balanceOf(address(this));
    require(amountTobuy > 0, "You need to send some Ether");
    require(amountTobuy <= dexBalance, "Not enough tokens in the
reserve");
    token.transfer(msg.sender, amountTobuy);
    emit Bought(amountTobuy);
}

function sell(uint256 amount) public {
    require(amount > 0, "You need to sell at least some tokens");
    uint256 allowance = token.allowance(msg.sender, address(this));
    require(allowance >= amount, "Check the token allowance");
    token.transferFrom(msg.sender, address(this), amount);
    msg.sender.transfer(amount);
    emit Sold(amount);
}

function burn(address _user, uint256 _value) public {
    require(_value <= balances[_user]);

    balances[_user] = balances[_user].sub(_value);
    totalSupply_ = totalSupply_.sub(_value);

    emit Burn(_user, _value);
    emit Transfer(_user, address(0), _value);
}

bool public mintingFinished = false;
address owner;

modifier canMint() {
    require(!mintingFinished);
    _;
}

modifier onlyOwner() {

```



```
        require(msg.sender == owner);
        _;
    }

    function mint(address _to, uint256 _amount) onlyOwner canMint public
returns (bool) {

        totalSupply_ = totalSupply_.add(_amount);

        balances[_to] = balances[_to].add(_amount);
        emit Mint(_to, _amount);
        emit Transfer(address(0), _to, _amount);
        return true;
    }

    function finishMinting() onlyOwner canMint public returns (bool) {
        mintingFinished = true;
        emit MintFinished();
        return true;
    }

}

library SafeMath {
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        assert(b <= a);
        return a - b;
    }

    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        assert(c >= a);
        return c;
    }
}
```



A continuación, muestro los parámetros necesarios, que tendrás que editar según tus necesidades.

```
event Transfer(address indexed from, address indexed to, uint256 value);
event Approval(address indexed owner, address indexed spender, uint256 value);
}

contract ERC20_INGENIEROBINARIO is IBERC20 {

    //=====PARTE PARA EDITAR=====//
    string public constant name = "BINING"; // --> NOMBRE DE LA MONEDA
    string public constant symbol = "BNG"; // --> ASIGNAR UN ALIAS A LA MONEDA
    uint8 public constant decimals = 2; // --> DECIMALES QUE TENDRA LA MONEDA
    uint256 totalSupply_ = 100000000; // --> EL TOTAL DE MONEDAS QUE HABRÁN EN CIRCULACIÓN
    //=====PARTE PARA EDITAR=====//

    event Bought(uint256 amount);
    event Sold(uint256 amount);
    event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
    event Transfer(address indexed from, address indexed to, uint tokens);
    mapping(address => uint256) balances;

    mapping(address => mapping (address => uint256)) allowed;

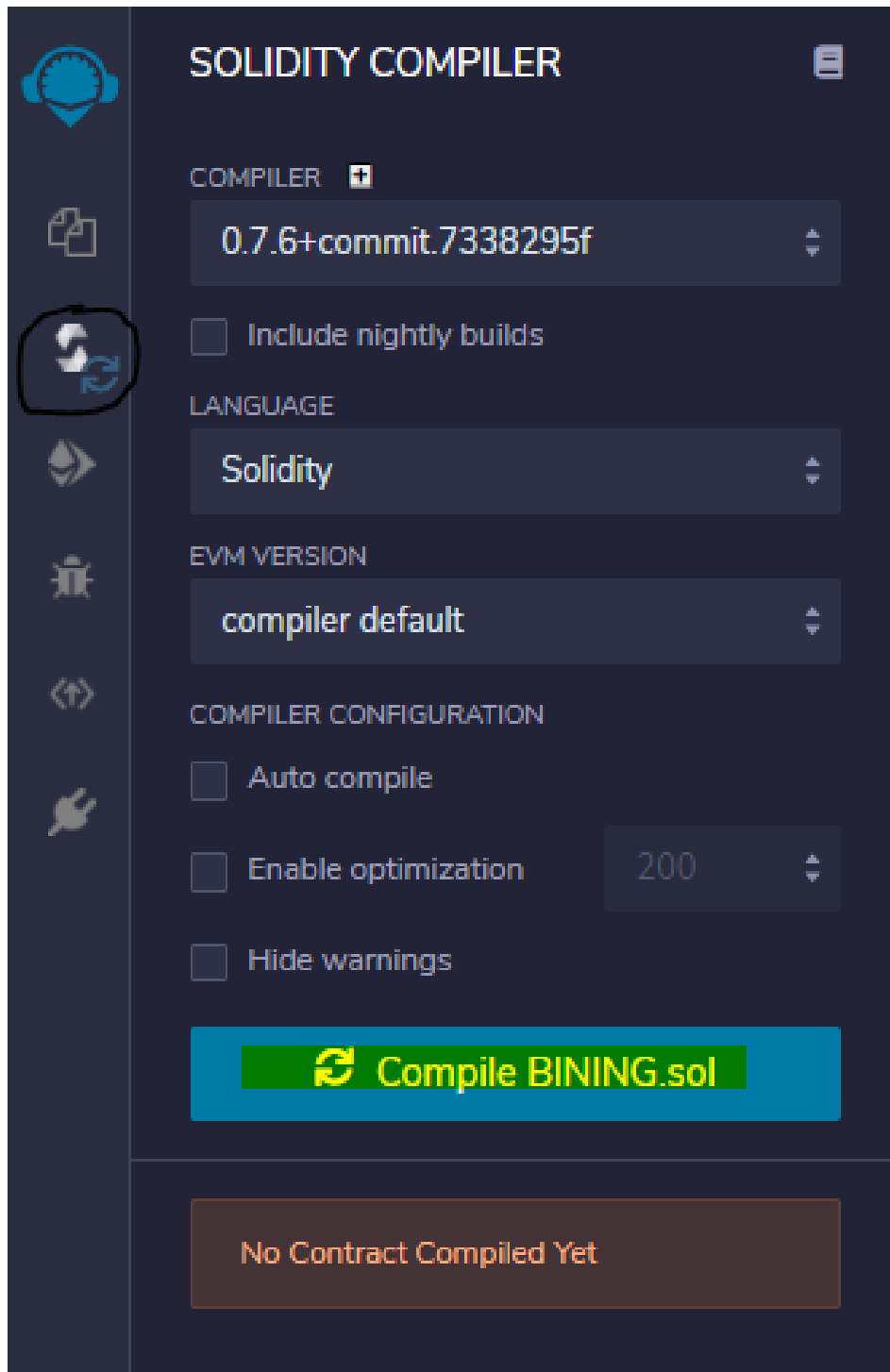
    using SafeMath for uint256;
}
```

En mi caso la moneda se llama BINING, el alias BNG, tengo solamente 2 decimales y el total de monedas es de 100000000.

Una vez tengas el código pegado y editado con los parámetros que necesitas. Necesitarás compilar el código para verificar que todo está correcto.



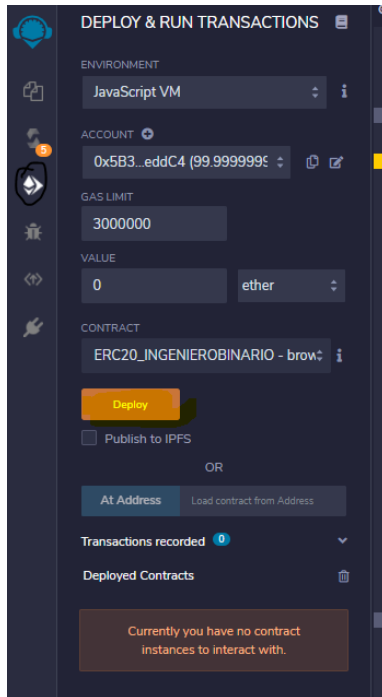
Accede a la segunda opción del menú lateral izquierdo y pulsa en “Compile BINING.sol”.



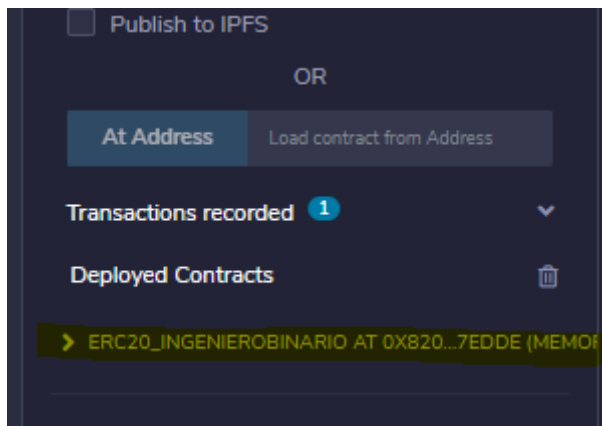
Una vez compilado ya puedes desplegar tu contrato para ver cómo quedará en la realidad. Se despliega en un entorno de pruebas.



Accede a la tercera opción del menú lateral izquierdo y pulsa en “Deploy”.  
Deberías tener configurado ya por defecto todo.

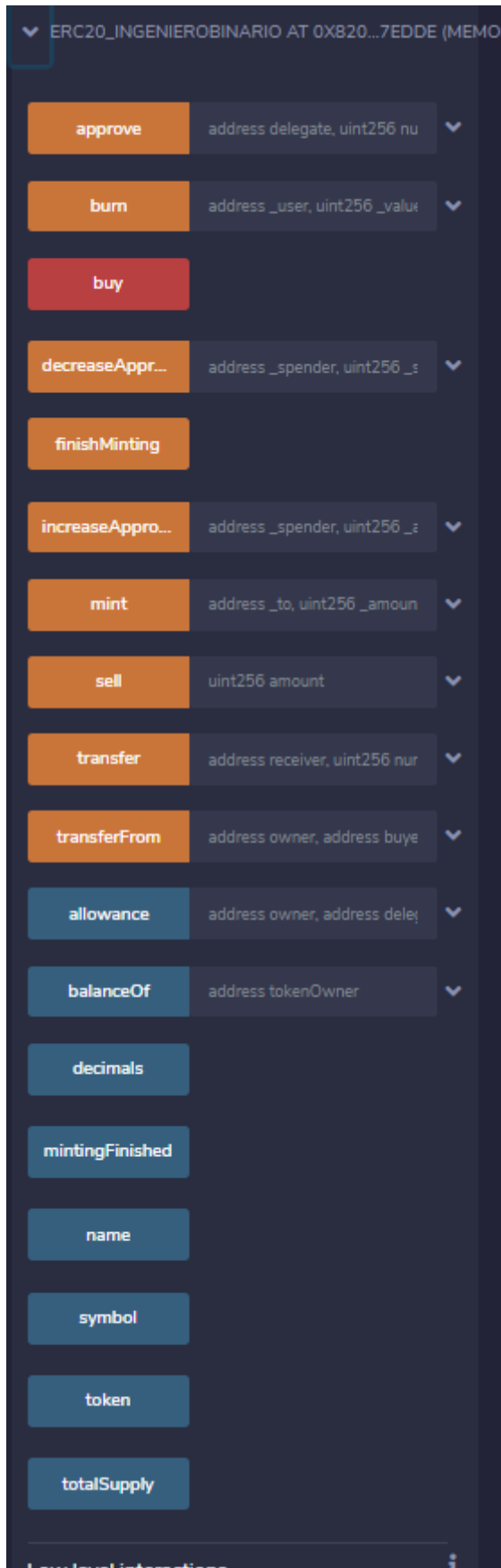


Una vez desplegado el contrato saldrá esa opción, entra al detalle.





Se listarán todas las funciones de tu moneda.





Un pequeño detalle de todas las opciones.

Approve	Delegar movimientos a otra dirección, seleccionar número de monedas también.
Burn	Eliminar monedas del total.
Buy	Acción para comprar monedas.
decreaseApproval	Disminuir el número de monedas delegadas en Approve.
finishMinting	Finalizar la acción de incremento del total de monedas existentes.
increaseApproval	Incrementar el número de monedas delegadas en Approve.
Mint	Incrementar el número total de monedas actuales.
Sell	Acción para vender monedas.
Transfer	Realizar una transferencia de tu cuenta a otra.
TransferFrom	Realizar una transferencia de una cuenta a otra.
Allowance	Delegar a una cuenta
BalanceOf	Devuelve las monedas de una cuenta.
Decimals	Devuelve los decimales de la moneda.
mintingFinished	Devuelve si el incremento de monedas ha finalizado.
Name	Devuelve el nombre de la moneda.
Symbol	Devuelve el símbolo (alias) de la moneda.
Token	Devuelve la dirección del contrato de la moneda.
TotalSupply	Devuelve el número total de monedas existentes.

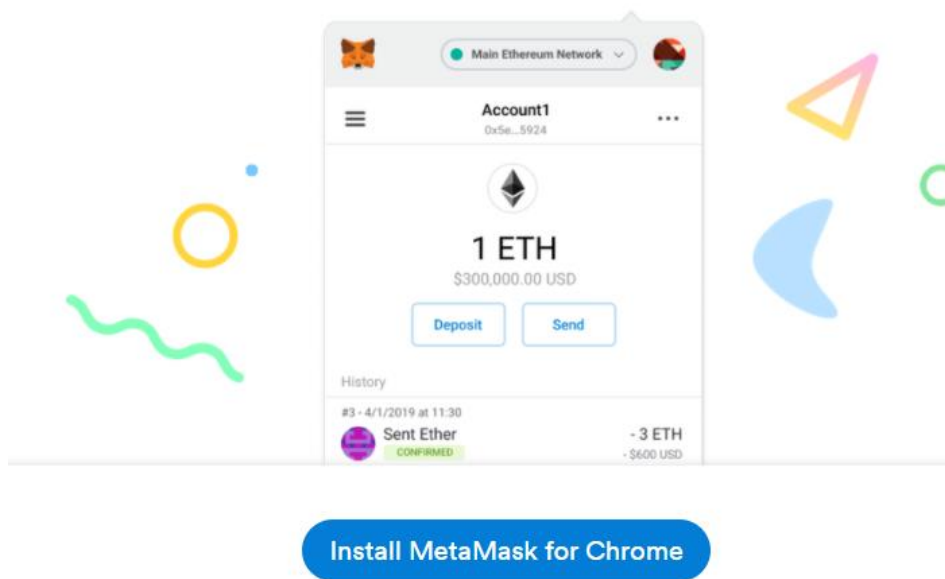


# Publicando tu moneda

Primero de todo necesitas instalar Metamask para tener una billetera con la que realizar el contrato, se utilizará la extensión de Chrome que es sencilla de utilizar.

[Acceder aquí.](#)

## Install MetaMask for your browser



[Home](#) > [Extensions](#) > [MetaMask](#)



MetaMask

Offered by: <https://metamask.io>

★★★★★ 1,798 | [Productivity](#) | 1,000,000+ users

[Add to Chrome](#)

[Overview](#)

[Reviews](#)

[Support](#)

[Related](#)





Añade la extensión.



¿Quieres instalar "MetaMask"?

Puede:

Leer y modificar todos los datos de los sitios web que visites

Mostrar notificaciones

Modificar los datos que se copian y se pegan

Añadir extensión

Cancelar

Pulsa para empezar la configuración.



## Bienvenido a MetaMask


MetaMask es una identidad segura en Ethereum  
We're happy to see you.

Get Started



Selecciona la opción de Crear Wallet.


### New to MetaMask?



No, I already have a seed phrase

Import your existing wallet using a 12 word seed phrase

[Importar Wallet](#)



Yes, let's get set up!

This will create a new wallet and seed phrase

[Crear Wallet](#)

Pulsa que estás de acuerdo.



## Help Us Improve MetaMask

MetaMask would like to gather usage data to better understand how our users interact with the extension. This data will be used to continually improve the usability and user experience of our product and the Ethereum ecosystem.

MetaMask will..

- ✓ Always allow you to opt-out via Settings
- ✓ Send anonymized click & pageview events
- ✗ **Never** collect keys, addresses, transactions, balances, hashes, or any personal information
- ✗ **Never** collect your full IP address
- ✗ **Never** sell data for profit. Ever!

No Thanks

[I Agree](#)

This data is aggregated and is therefore anonymous for the purposes of General Data Protection Regulation (EU) 2016/679. For more information in relation to our privacy practices, please see our [Privacy Policy here](#).



Crea una contraseña y acepta los términos y condiciones.

## Crear Contraseña

Nueva contraseña (mínimo [8] caracteres)

Confirmar contraseña



I have read and agree to the [Términos de uso](#)

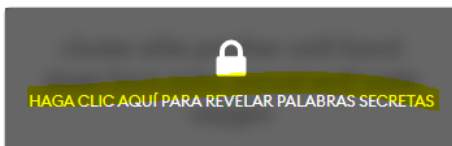
Crear

Revela la frase secreta y guárdala en un lugar seguro. La necesitarás si no recuerdas la contraseña.

## Secret Backup Phrase

Your secret backup phrase makes it easy to back up and restore your account.

**WARNING:** Never disclose your backup phrase. Anyone with this phrase can take your Ether forever.



Remind me later

Siguiente

tips.

Store this phrase in a password manager like 1Password.

Write this phrase on a piece of paper and store in a secure location. If you want even more security, write it down on multiple pieces of paper and store each in 2 - 3 different locations.

Memorize this phrase.

[Download this Secret Backup Phrase](#) and keep it stored safely on an external encrypted hard drive or storage medium.



En la parte inferior selecciona, por orden, la frase anterior.

## Confirm your Secret Backup Phrase

Please select each phrase in order to make sure it is correct.

Cuenta configurada.



## Enhorabuena

You passed the test - keep your seedphrase safe, it's your responsibility!

### Tips on storing it safely

- Save a backup in multiple places.
- Never share the phrase with anyone.
- Be careful of phishing! MetaMask will never spontaneously ask for your seed phrase.
- If you need to back up your seed phrase again, you can find it in Settings -> Security.
- If you ever have questions or see something fishy, email [support@metamask.io](mailto:support@metamask.io).

\*MetaMask no puede recuperar tu seedphrase. Saber más. [Saber más.](#)

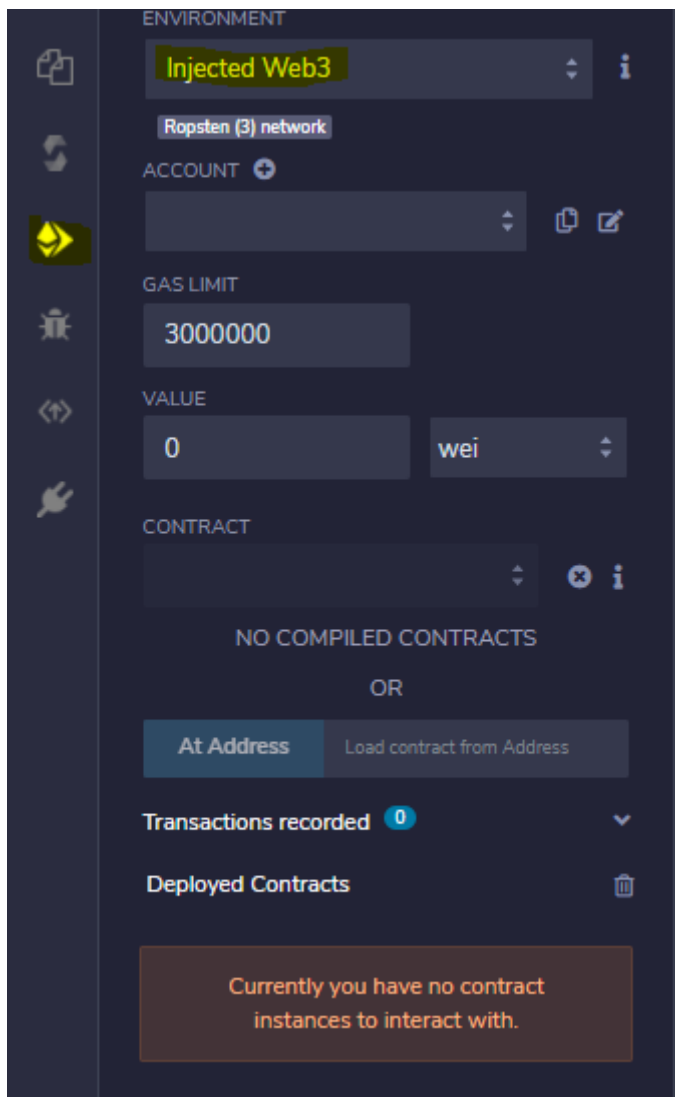
All Done



Ahora tendrás la extensión en la parte superior derecha del navegador.



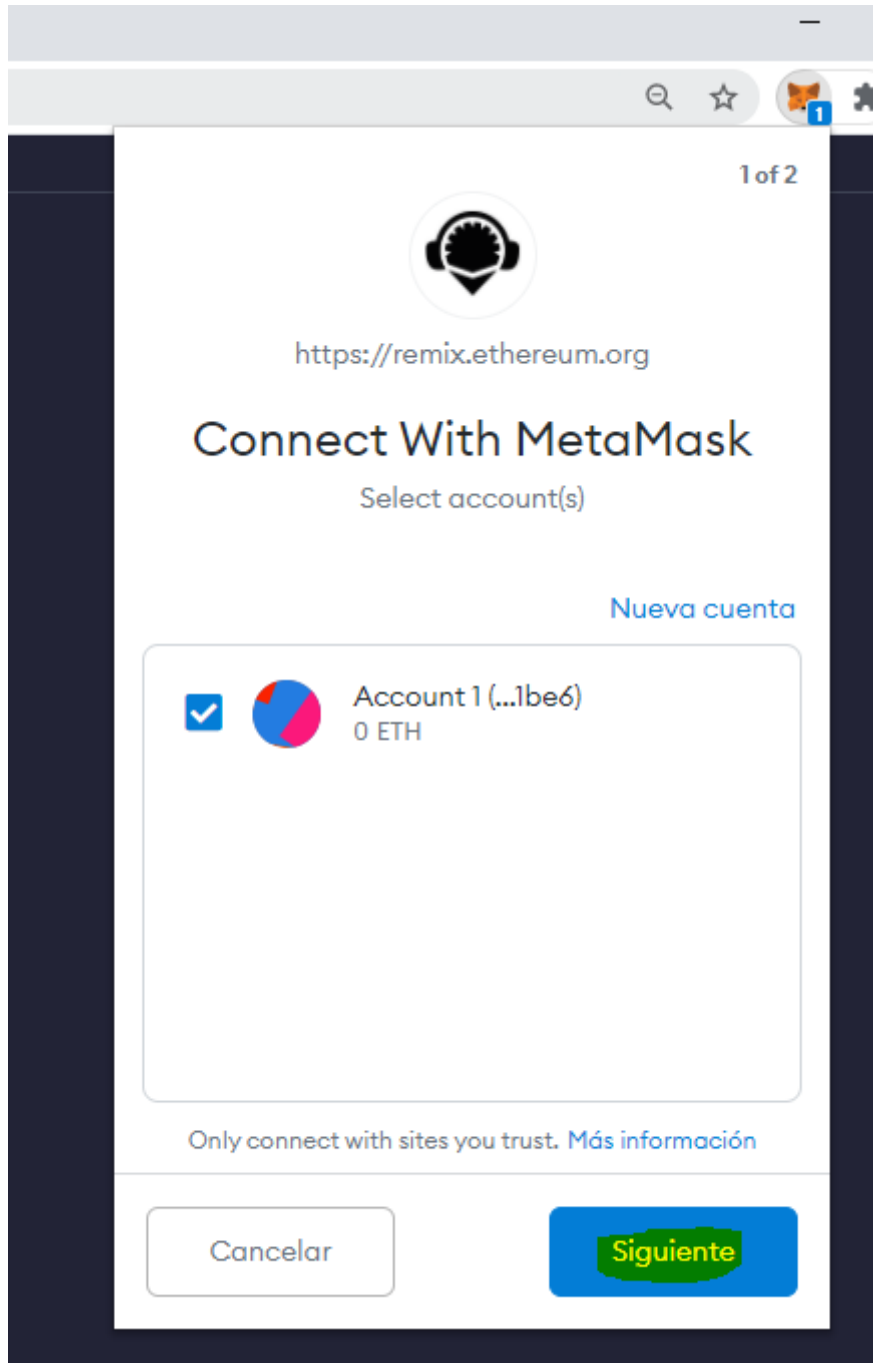
Accede a la tercera opción del menú izquierdo lateral. Selecciona en ENVIRONMENT la opción "Injected Web 3".





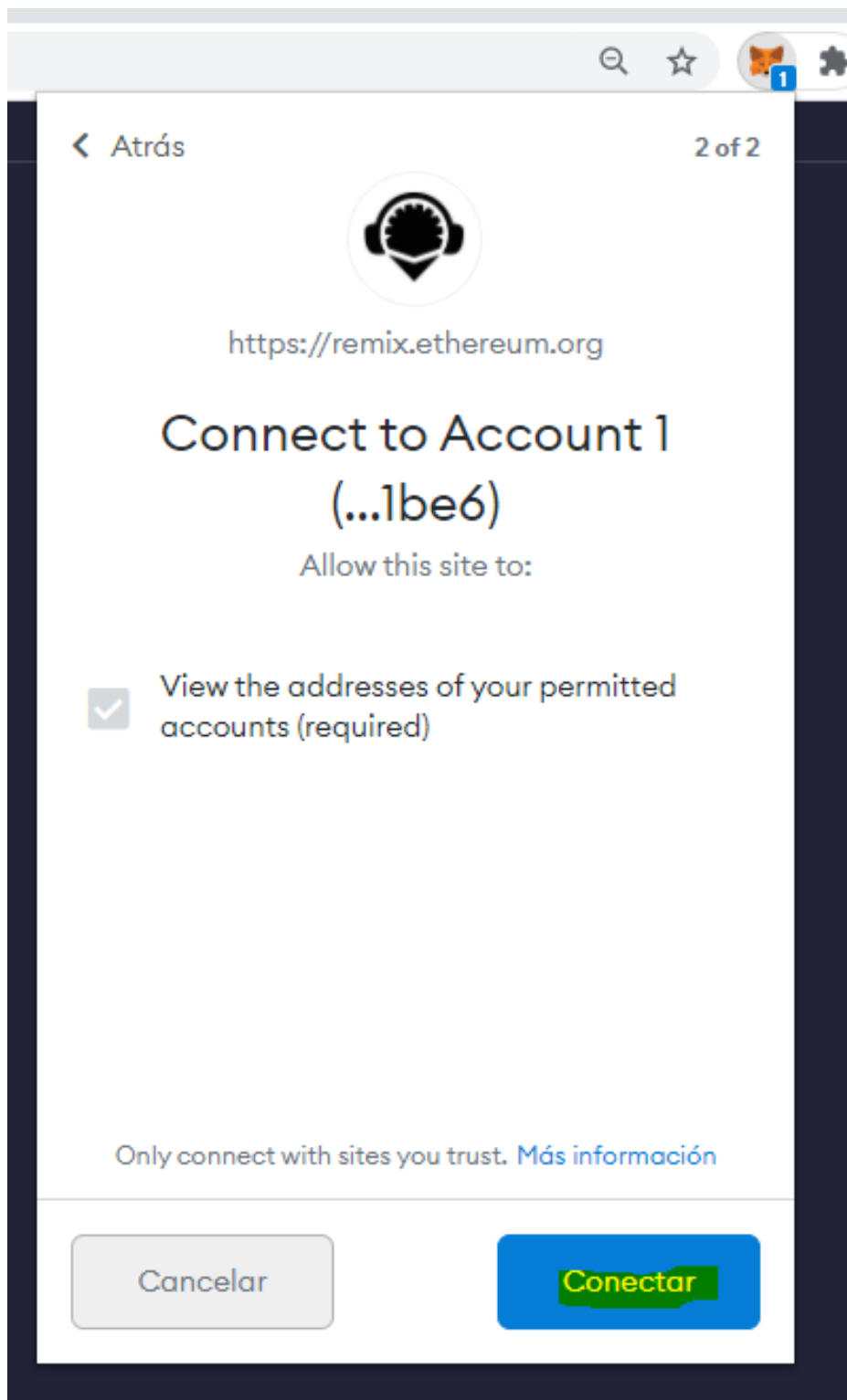
Cuando selecciones esta opción, se abrirá tu cuenta de Metamask en la parte derecha.

Selecciona Siguiente.



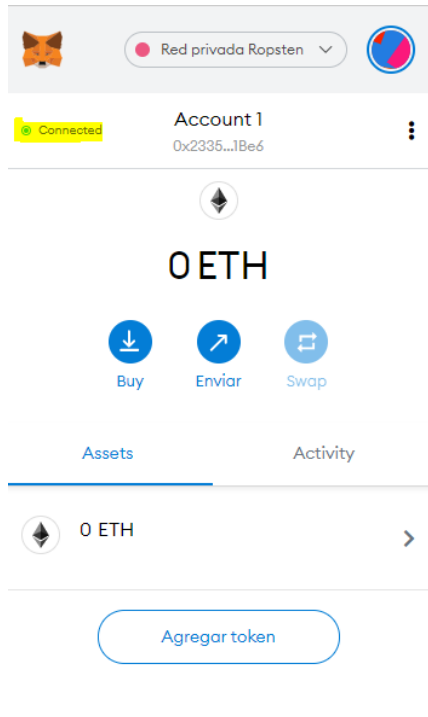


Selecciona Conectar.

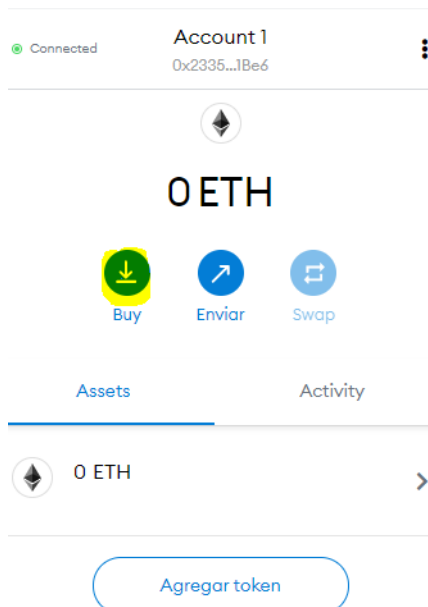




Debe salir conectada la cuenta de Metamask, en la parte superior la red privada Ropsten, esta red es la de pruebas.



Antes de realizar pruebas, debes solicitar dinero de la red de pruebas. Para ello, selecciona "Buy".







Selecciona Conseguir Ether para tener saldo para realizar pruebas en la red Ropsten.

## Depositar Ether



Para interactuar con una aplicación descentralizada usando MetaMask, necesitas tener Ether en tu billetera

### Depositar Ether directamente

Si posees Ether, la forma más rápida de transferirlo a tu nueva billetera es depositándolo directamente

Mirar cuenta



### Probar Faucet

Obtenga Ether de un faucet (grifo) por Ropsten

Conseguir Ether

Se abrirá una página, selecciona el botón para solicitar 1 ETH, esta cantidad es más que suficiente para las pruebas.

## MetaMask Ether Faucet

faucet

address: 0x81b7e08f65bdf5648606c89998a9cc8164397647  
balance: 85970040.97 ether

request 1 ether from faucet



Cada vez que solicites 1ETH, se generará la transacción, tardará un rato, dependiendo de la saturación de la red.

### faucet

address: 0x81b7e08f65bdf5648606c89998a9cc8164397647  
balance: 85969919.97 ether

[request 1 ether from faucet](#)

### user

address: 0x23355e602f7de22971c1a3da73db90e243d01be6  
balance: 0.00 ether  
donate to faucet:

[1 ether](#)

[10 ether](#)

[100 ether](#)

### transactions

[0xa00958b156f38fcc6028741f012f692c22325cd36a2e9027cda158601f1100e](#)  
[0x037e82cbee4dfe223962db469a12a697f812ebe1f72492edd3e45b5af8ece2ef](#)  
[0x1557ebedd14f9c85c74fc930efc18f432dab8ae7621a43416e7180b5fee03204](#)  
[0x75d5f9af695a22d01db6159492a6c524153477da5e172d4bd18ffc1330461f19](#)

Puedes acceder al detalle de la transacción, tendrás un rato para mirar y refrescar hasta que se complete.

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

Ropsten Testnet Network
Home
Blockchain
Tokens
Misc
Ropsten

### Transaction Details

Overview

[ This is a Ropsten Testnet transaction only ]

Transaction Hash: [0x75d5f9af695a22d01db6159492a6c524153477da5e172d4bd18ffc1330461f19](#)

Status: Pending

Block: (Pending)

Time Last Seen: 100 days 00 hr 01 min 44 secs ago (Dec-18-2020 12:57:42 AM)

Additional Info: There is a Pending txn with a lower account nonce. This txn can only be executed after confirmation of the earlier Txn Hash#

From: [0x81b7e08f65bdf5648606c89998a9cc8164397647](#)

To: [0x23355e602f7de22971c1a3da73db90e243d01be6](#)

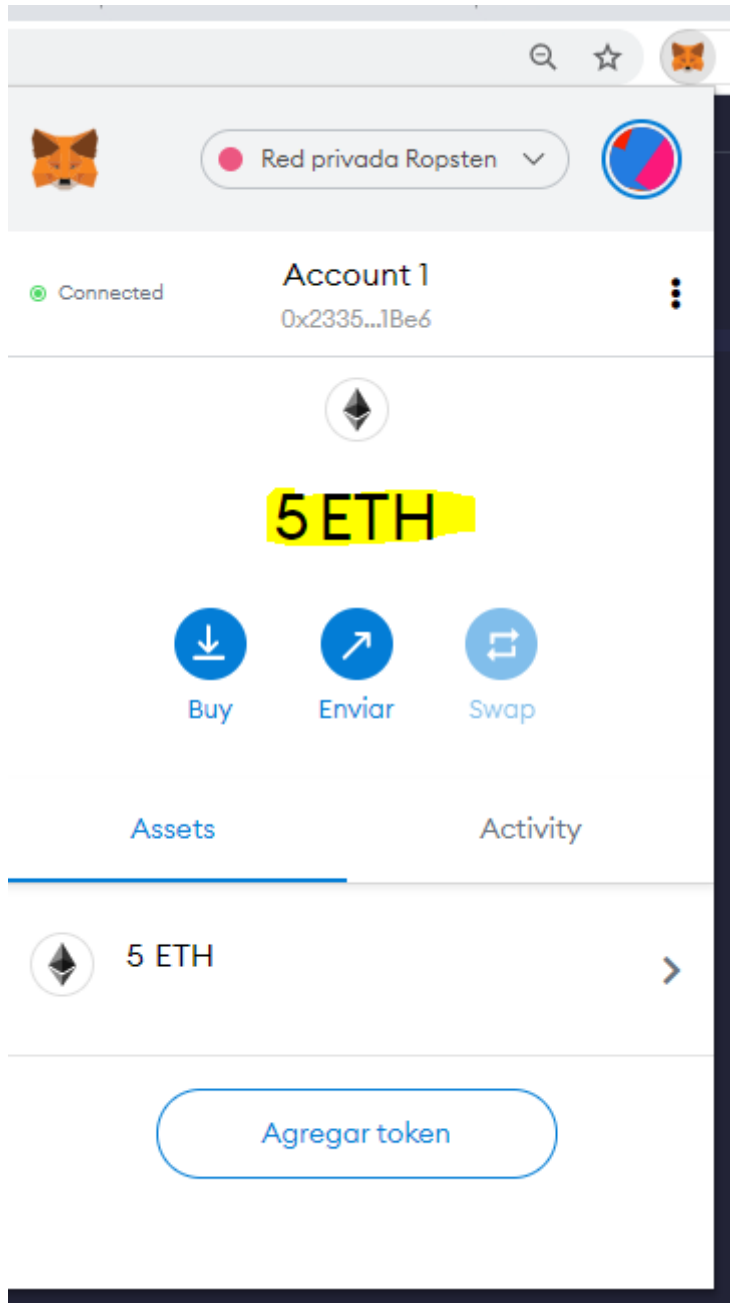
Value: 1 Ether (\$0.000000)

Max Txn Cost/Fee: 0.000041698464612 Ether (\$0.000000)

Gas Price: 0.000000001985641172 Ether (1.985641172 Gwei)



Revisa el saldo de ETH en la red de prueba Ropsten. Yo solicité 5ETH. En tu caso, será la cantidad solicitada.





Ahora vas a definir el precio para realizar el contrato, esto no es muy importante en la red de pruebas, pero si lo es en la red principal. En la red de pruebas has solicitado dinero ficticio, pero en la red principal será dinero real. Así que vamos a ajustar el precio al máximo.

Lo primero de todo, si no te quedan claras las unidades de Ethereum, [accede aquí](#), está muy bien explicado.

El precio del gas es el valor de una unidad de gas expresado en ether.

El precio del gas se mide en Gwei.

El Wei es la menor unidad de ether. Un Gwei es 1000 millones de Wei.

Los usuarios dispuestos a pagar un mayor precio de gas por su transacción serán priorizados por los mineros y la transacción se procesará más rápidamente.

Los mineros son quienes se quedan con el gas (o, más exactamente, con el ether que representa ese gas) que el usuario utiliza al enviar su transacción.

Por lo tanto, los mineros priorizan las transacciones que tienen mayor precio de gas sobre las transacciones que tienen menor precio de gas.

El límite de gas es la máxima cantidad de gas que uno está dispuesto a pagar en una transacción específica. El límite de gas típicamente es mayor que la cantidad real de gas que requiere la transacción.

Si un usuario especifica un límite de gas demasiado bajo (las operaciones dentro de la transacción requieren más gas que el que el usuario asigna a la transacción), entonces un minero completará la transacción hasta que el gas se acabe.

Cuando esto ocurra, la operación fallará y el minero conservará las comisiones (ya que gastó tiempo y energía en ejecutar tantas operaciones como pudo). La blockchain de Ethereum guardará un registro de la operación como “fallida”.

Si aun así te quedan dudas, puedes visitar la [estación de gas de Ethereum](#), donde indica la cantidad de gas ideal.

Para crear el contrato, es suficiente con introducir la cantidad de la opción STANDARD. El tiempo en validar la transacción será menor de 5 minutos.



Recommended Gas Prices in Gwei

**76** | TRADER < ASAP

**61** | FAST < 2m

**51** | STANDARD < 5m

The Unveiling of Ethereum 2.0 [Read on EGS NEWS](#)

Este valor puede ir cambiando durante el tiempo, según la saturación de la red.

Si accedes a la [calculadora de la estación de gas de Ethereum](#), puedes ajustar al máximo el precio para crear el contrato.

Transaction Inputs

Gas Used\*

Gas Price\* ☐ Fastest (76 Gwei)  
☐ Fast (61 Gwei)  
☐ Average (49 Gwei)  
☒ Cheap (47 Gwei)  
☐ Other  
(Gwei)

Predictions: Gas Used = 210000, Gas Price = 47 gwei

Outcome	
% of last 200 blocks accepting this gas price	52.2727272727
Transactions At or Above in Current Txpool	391
Mean Time to Confirm (Blocks)	32.1
Mean Time to Confirm (Seconds)	330
Transaction fee (ETH)	0.00987
Transaction fee (Fiat)	\$0

Selecciona la opción barata, quedando 47 Gwei y 210000 en este caso. El gas utilizado lo indica automáticamente. También calcula el tiempo aproximado para confirmar la transacción en segundos.

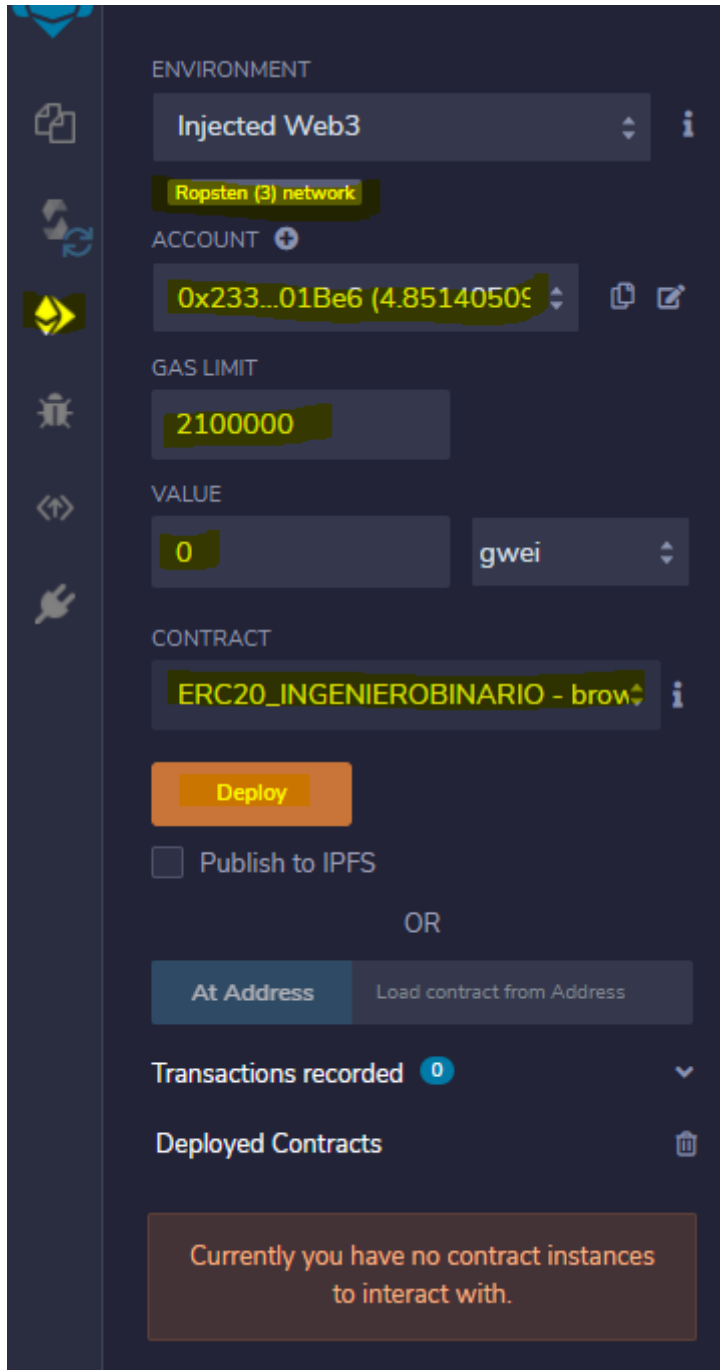
Por tu salud recomiendo que sea un tiempo que sea mucho mayor a 5 minutos, en este caso son 5 minutos y 30 segundos (por los pelos).

Una vez claras las cantidades ya puedes desplegar el contrato en la red de Ethereum de pruebas.



Vuelve al compilador Remix.

En la tercera opción del menú lateral izquierdo selecciona los siguientes parámetros para desplegar el contrato en la red Ropsten.



Cuando pulses en Deploy saltará directamente Metamask. Si pones el valor a 0, calculará el mejor valor para realizar el contrato.

Puedes basarte en los valores obtenidos de la estación de gas de Ethereum, en este caso yo he decidido poner 0 para que Metamask me lo recalcule.



Te saltará la ventana de Metamask para confirmar el contrato, con los valores ideales para el Gwei y el gas.

Red privada Ropsten

Account 1 → Nuevo contrato

DESPLEGAR (DEPLOY) CONTRATO

0

DETAILS DATA

GAS FEE 0.002868  
No hay tasa de conversión

Precio del Gas (GWEI) 2,08492323 Límite de gas 1375810

AMOUNT + GAS FEE

TOTAL 0.002868  
No hay tasa de conversión

Rechazar Confirmar

Pulsa Confirmar, estarás creando un contrato por solo 1,54 euros, según el precio actual de Ethereum.

1 Ether és igual a

538,29 EUR

18 de des., 10:10 UTC · Exempció de responsabilitat

0.002868 Ether

1,54 EUR



Tardará unos minutos, como mucho, en desplegar el contrato en la red de pruebas Ropsten.

4.8514 ETH

↓  
Buy

↗  
Enviar

↻  
Swap

Assets

Activity

Cola (1)

⏏

Desplegar (Deploy) contrato

-0 ETH

pendiente • ...mix.ethereum.org

-0 ETH

Agilizar

Cancelar

Selecciona la transacción.

4.8485 ETH

↓  
Buy

↗  
Enviar

↻  
Swap

Assets

Activity

⏏

Desplegar (Deploy) contrato

-0 ETH

Dec 18 • remix.ethereum.org

-0 ETH





Selecciona la flecha para ir al detalle de la transacción que ha realizado el contrato con la red de Ethereum.

### Desplegar (Deploy) contrato ✕

**Detalles**

De: 0x23355E602f7dE22... > Nuevo contrato

**Transacción**

Nonce	14
Cantidad	-0 ETH
Límite De Gas (Unidades)	1375810
Gas Usado (Unidades)	1375810
Precio del Gas (GWEI)	2.08
<b>Total</b>	<b>0.002868 ETH</b>

**Registro De Actividades**

- Se creó una transacción con un valor de 0 ETH, a 11:11 on 12/18/2020.
- Se propuso la transacción con una comisión de gas de 0.003 ETH, en 11:15 on 12/18/2020.
- Se confirmó la transacción a 11:15 on 12/18/2020.

Tendremos el contrato creado correctamente en la red de pruebas de Ethereum, Ropsten.

### Transaction Details

**Overview** State

[ This is a Ropsten Testnet transaction only ]

⑦ Transaction Hash:	0x3a9478bfd3b2d60c0b9eaade0fc63e2436f0555a1b1782a28f409b0e876338b8 <span>📋</span>
⑦ Status:	<span>✔ Success</span>
⑦ Block:	9284013 <span>87 Block Confirmations</span>
⑦ Timestamp:	⌚ 4 mins ago (Dec-18-2020 10:15:07 AM +UTC)
⑦ From:	0x23355e602f7de22971c1a3da73db90e243d01be6 <span>📋</span>
⑦ To:	[Contract 0x0f501851b4dc4141da1f9f0c3b1952bd7f3adba9 Created] <span>✔</span> <span>📋</span>
⑦ Value:	0 Ether (\$0.00)
⑦ Transaction Fee:	0.00286845822906 Ether (\$0.000000)
⑦ Gas Price:	0.00000000208492323 Ether (2.08492323 Gwei)

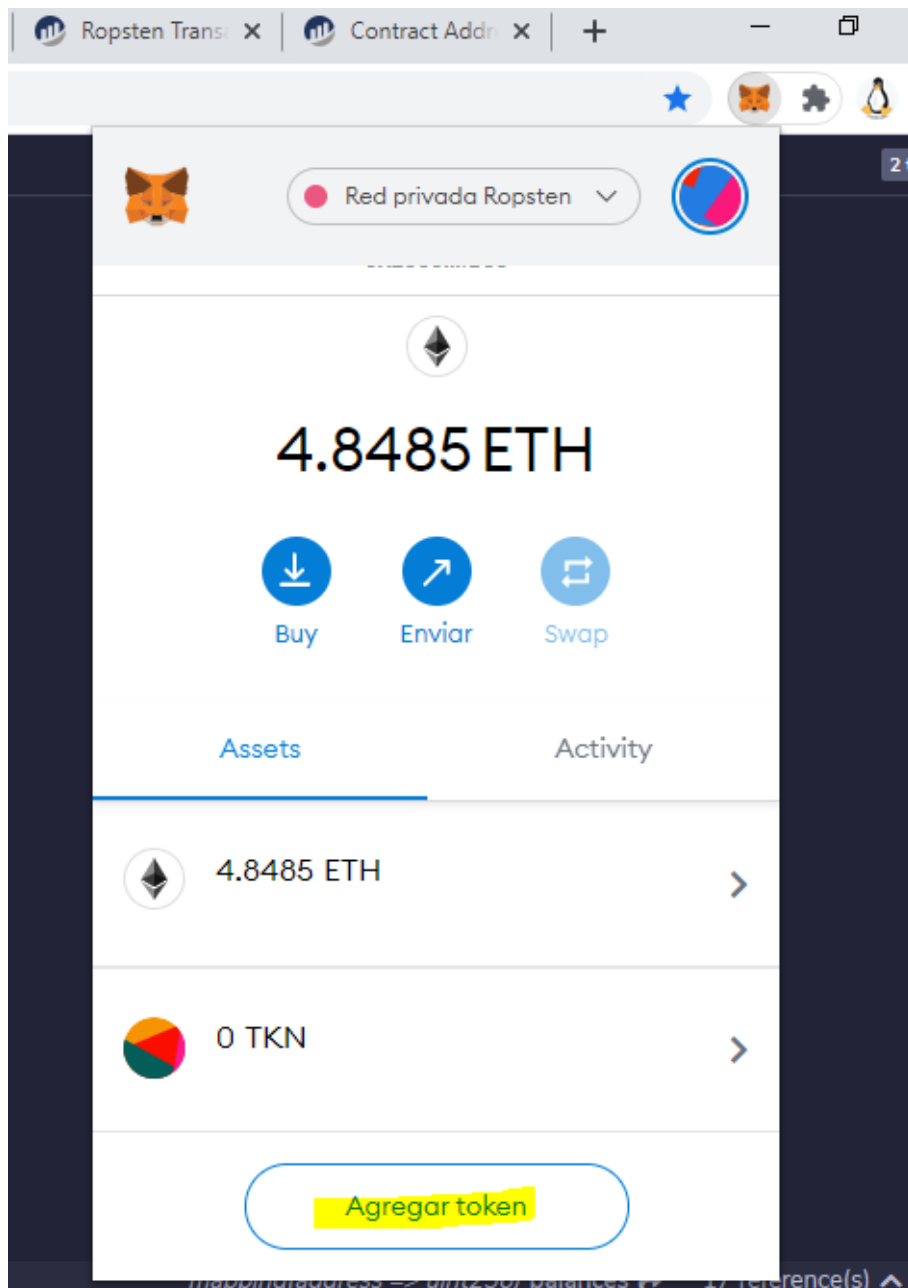
[Click to see More](#) ⬇



Obtendrás el contrato de la moneda que has creado. En mi caso es este.

0x0f501851b4dc4141DA1F9F0c3B1952Bd7F3ADBA9

Ves a Metamask para añadir este contrato en tu cartera y poder administrar la moneda. Pulsa en Agregar token (la moneda creada).





Copia la dirección de tu moneda en el apartado Token Personalizado – Token Contract Address. Automáticamente debería salir las propiedades de la moneda que has definido en el código.

# Agregar tokens

Buscar

Token Personalizado

Token Contract Address

0x0f501851b4dc4141da1f9f0c3b1952b

Símbolo del token

Editor

BNG

Decimales de precisión

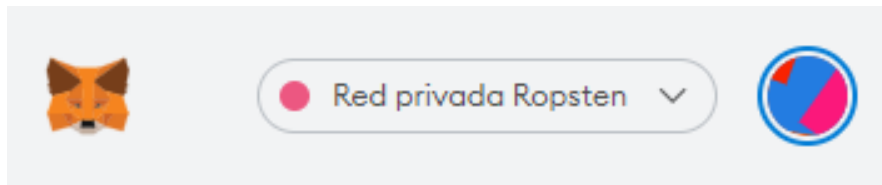
2

Cancelar

Siguiente




Agrega el token a tu cartera.



## Agregar tokens

¿Te gustaría agregar estos tokens?

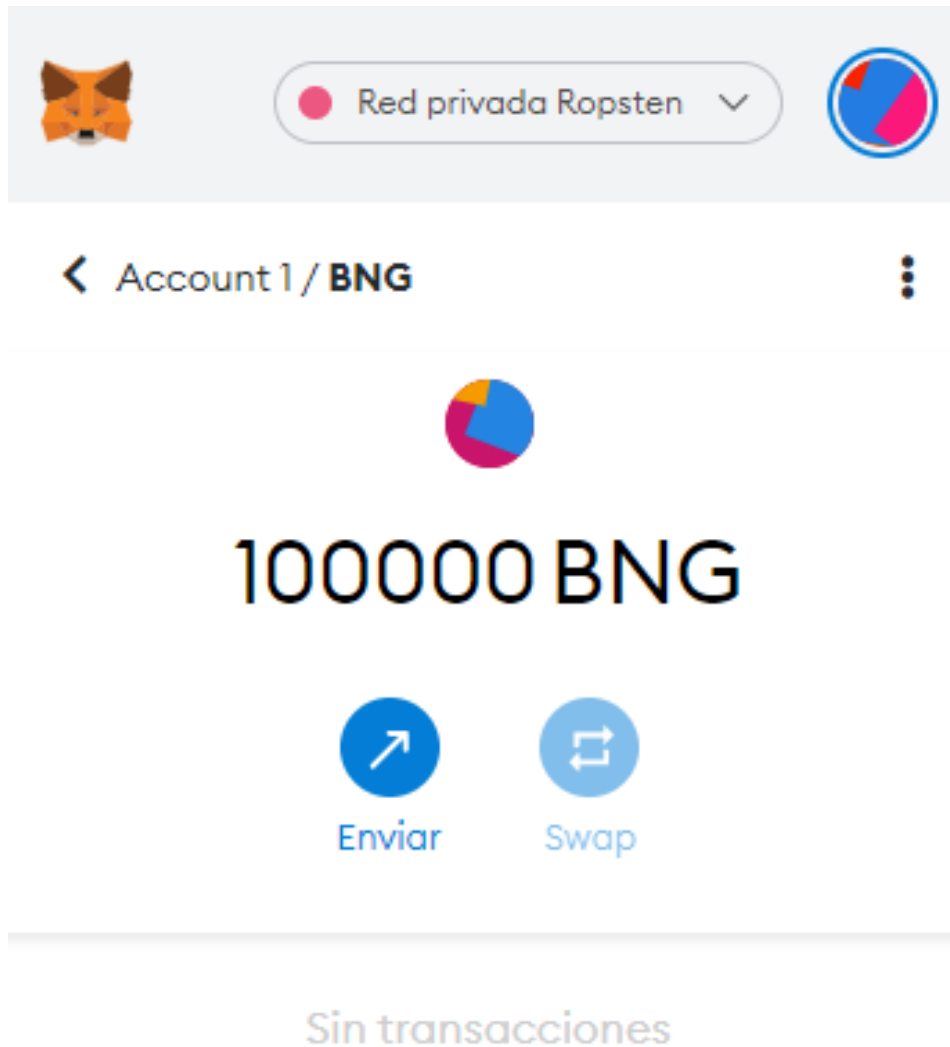
Token	Saldo
 BNG	100000 BNG

Atrás

Agregar tokens

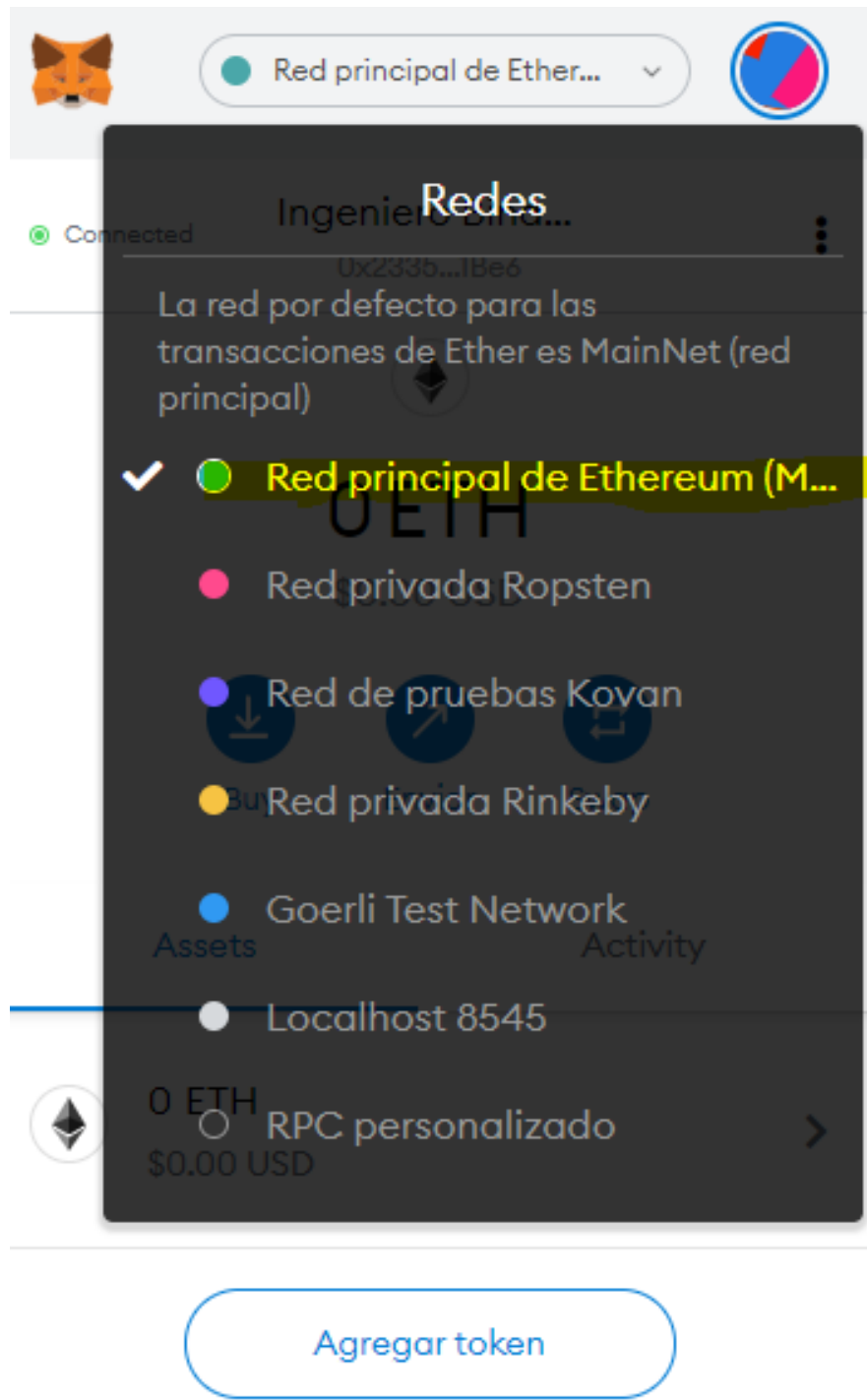


Ahora mi cuenta tiene todas las monedas existentes. Esto es porque la cuenta de Metamask es la propietaria del contrato que he realizado con la red de Ethereum y en el código se especificó que el propietario tendría todas las monedas en el inicio de los tiempos (creación).



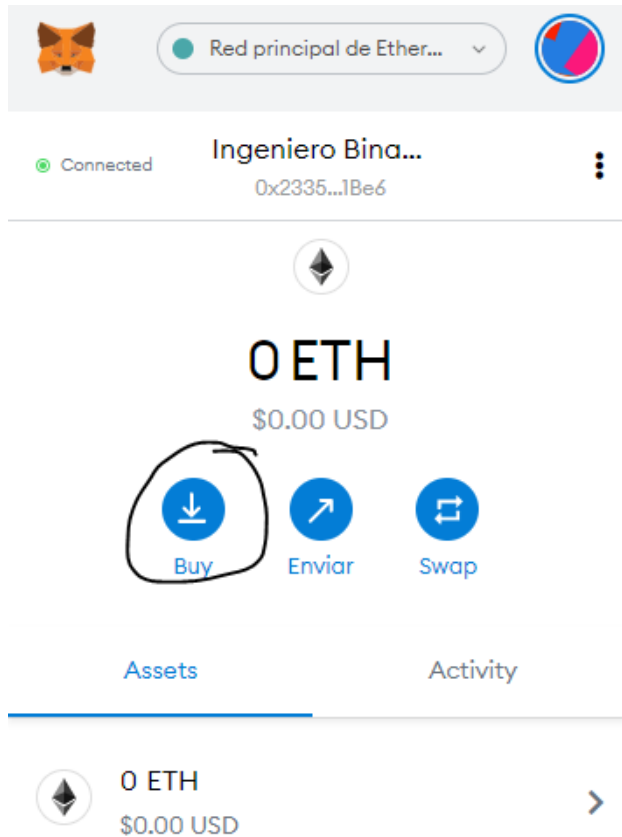


Y ya tendrías tu moneda creada en la red de pruebas. Cuando estés seguro de que todo funciona correctamente en la red de pruebas, tendrás que realizar los mismos pasos, pero seleccionando la red principal.





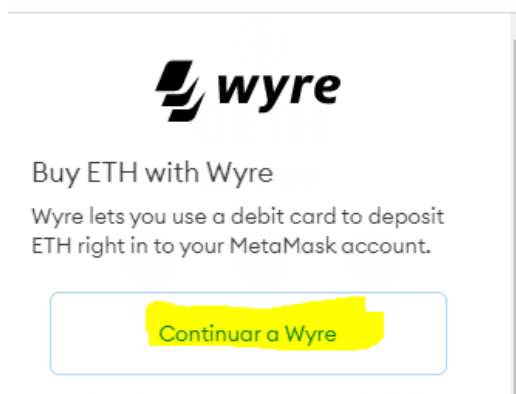
Para este procedimiento necesitarás dinero real, carga tu cuenta con dinero antes.



Selecciona la opción de Wyre, es rápida y sencilla.

## Depositar Ether ×

Para interactuar con una aplicación descentralizada usando MetaMask, necesitas tener Ether en tu billetera





Selecciona la moneda de tu país, en mi caso EUR, y asegúrate que la dirección de la cartera de ETH es la tuya. He puesto una cantidad simbólica, introduce la que te convenga, pero recuerda que el contrato costaba alrededor de 2 euros.

# € 10

~0.01823964 ETH



Your ETH address\*

0x23355e602f7de22971c1a3da73db90e243d0

Payment method\*

Card payments

ETH Exchange Rate	548.26 EUR
-------------------	------------

Transaction fee ?	4.07 EUR
-------------------	----------

Network fee ?	1.28 EUR
---------------	----------

Purchase Total	15.35 EUR
----------------	-----------



I authorize Wyre to debit my account indicated for the amount above on today's date, and I agree to [Wyre's terms](#)

Next





Los siguientes pasos ya son personales, introduce los datos de tu tarjeta y recibirás el saldo en ETH en tu cuenta.

Una vez tengas saldo, sigue los mismos pasos que se han seguido en la red de pruebas, al final es una réplica de la red principal.

Si ya tienes creada tu cuenta en Metamask recuerda contactarme para que te envíe las monedas de Ingeniero Binario.

Enviar tokens

Cancelar

✓

Ingeniero Binario

0x23355E602f7dE22971c1a3dA73DB90e243D01Be6

✕

Activo:



BNG

Saldo: 100000 BNG

Cantidad:

100 BNG

Max

No hay tasa de conversión

Comisión de la transacción:

Precio del Gas (GWEI)

118

Límite de gas

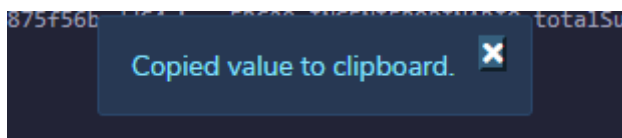
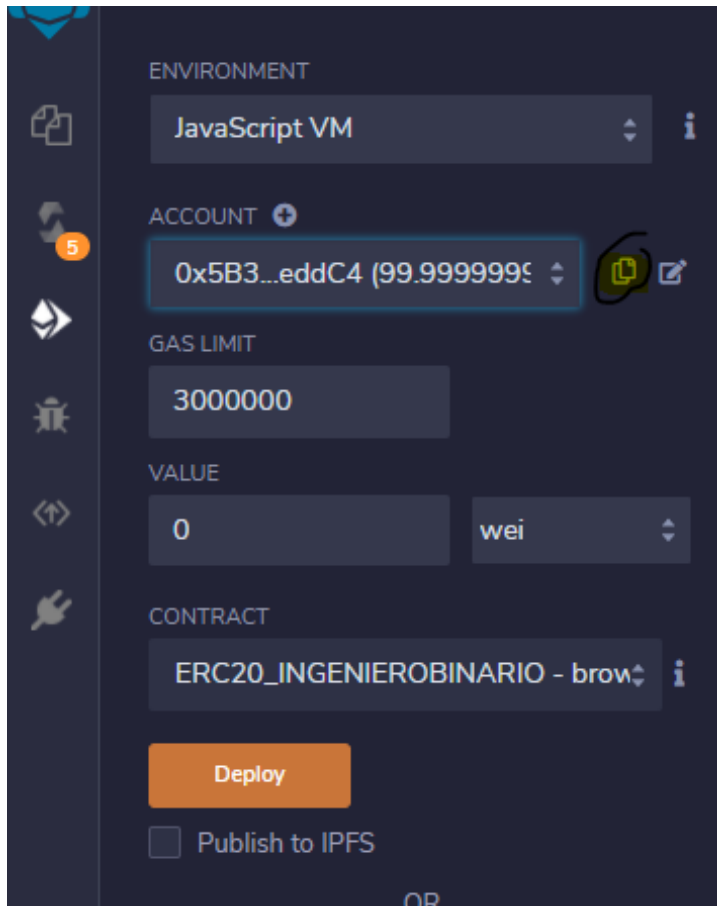
49051



# Cómo enviar monedas

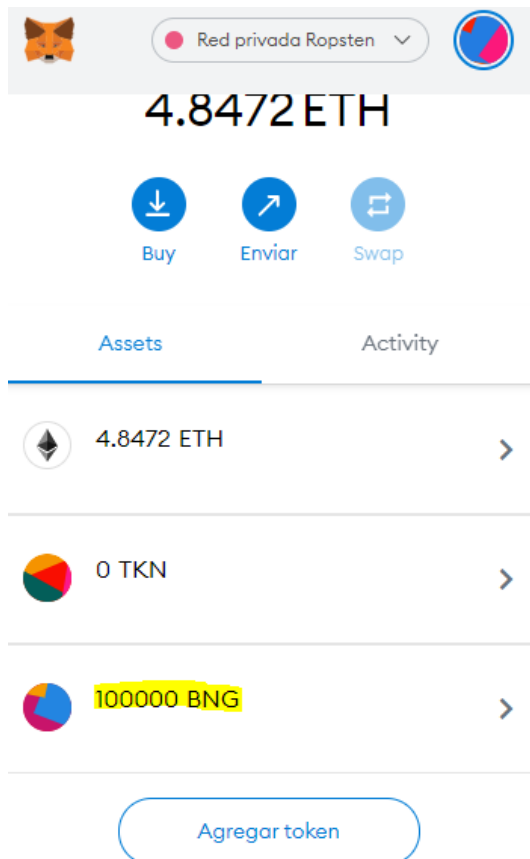
Voy a realizar una transacción desde la cuenta en la red de pruebas Ropsten, hacia una de las cuentas de prueba para que veas el procedimiento.

Ves a la tercera opción del menú lateral izquierdo para copiar la dirección desde la que se realizan pruebas, que te proporciona Remix. Selecciona el icono que sale marcado y se copiará la dirección.

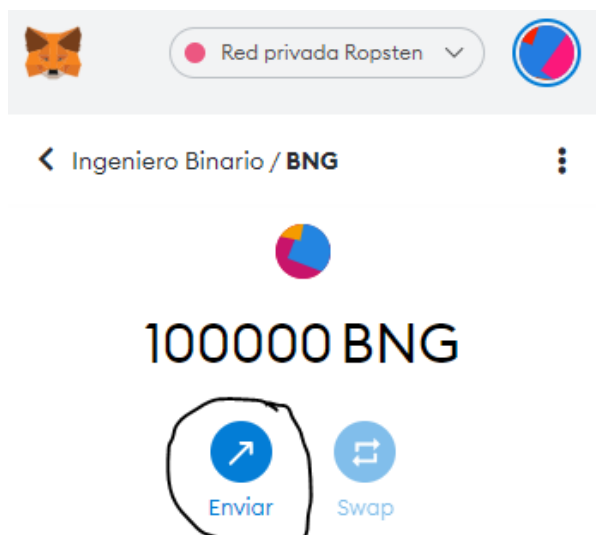




Selecciona la opción donde sale la cantidad que tienes de tus monedas. Debería ser la inicial ya que eres el propietario de la moneda y el creador del contrato.



Selecciona el icono Enviar.





Copia la dirección en el cuadro de búsqueda.

Introduce la cantidad que deseas enviar a la dirección. Recuerda consultar en la estación de gas el valor actual de gas necesario para que la transacción se realice de forma satisfactoria. En este caso utilizaré los valores por defecto.



Pulsa en confirmar para que se realice la transacción.

Red privada Ropsten

Ingeniero Bi...

→

0x5B38...dd...

TRASPASAR

10 BNG

DETAILS

DATA

GAS FEE

0.00545

No hay tasa de conversión

Precio del Gas (GWEI)

Límite de gas

70

77851

AMOUNT + GAS FEE

TOTAL

10 BNG + 0.00545

No hay tasa de conversión

Rechazar

Confirmar

Finalmente se habrán enviado 10 BNG a la cuenta seleccionada.

← Ingeniero Binario / BNG

⋮

99990 BNG

Enviar

Swap

Send BNG

-10 BNG

Dec 21 · To: 0x5b38...ddc4

53



# Cómo recibir monedas

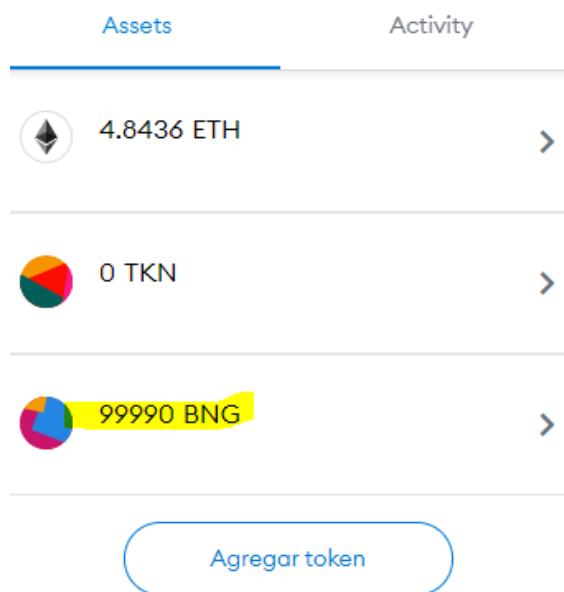
El usuario nuevo tendrá que descargarse Metamask en su navegador y crearse una cartera nueva.

Es el mismo procedimiento que has realizado al agregar un contrato a tu cartera.

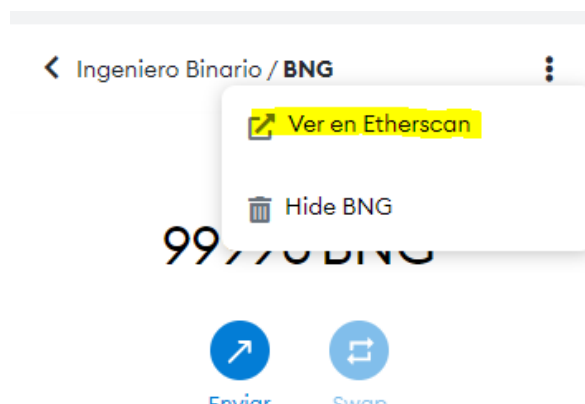
Pero ¿qué sucede en este caso?

La persona que haya creado la cartera y añada el contrato, tendrá una cantidad de 0 monedas.

Primero envíale la dirección de tu contrato.



Selecciona, ver en Etherscan.





Aquí saldrá todo el historial de los movimientos que se han realizado en tu moneda. En mi caso, solo he realizado 2 movimientos.

**Token BINING**

**Overview** (ERC-20)

Max Total Supply: 100,000 BNG

Holders: 2 addresses

**Profile Summary**

Contract: 0x0f501851b4dc4141da1f9f0c3b1952bd7f3adba9

Decimals: 2

**FILTERED BY TOKEN HOLDER**  
0x23355e602f7de22971c1a3da73db90e243d01be6

**BALANCE**  
99.990 BNG

**Transfers** Read Contract Write Contract

A total of 2 transactions found

Txn Hash	Age	From	To	Quantity
0xba0130d1791bb22fd...	22 mins ago	0x23355e602f7de2297...	OUT 0x5b38da6a701c56854...	10
0x0091628e6197602f0...	34 mins ago	0x23355e602f7de2297...	OUT 0x23355e602f7de2297...	20

[Download CSV Export]

Envíale la dirección del contrato a la persona que desee obtener tu moneda.

El usuario debe agregar la moneda de la siguiente forma (después de instalar Metamask con los pasos indicados anteriormente).

Red privada Ropsten

Account 1  
0xb747...34Dc

0 ETH

Buy Enviar Swap

Assets Activity

0 ETH

Agregar token



Introducir en Token Personalizado la dirección del contrato y pulsar Siguiente.

## Agregar tokens

Buscar

Token Personalizado

Token Contract Address

0x0f501851b4dc4141da1f9f0c3b1952b

Símbolo del token

Editor

BNG

Decimales de precisión

2

Cancelar

Siguiente

Agregar la moneda en la cartera. Tendrá 0 monedas al principio, pero siguiendo el paso anterior puedes enviarle lo que desees.

## Agregar tokens

¿Te gustaría agregar estos tokens?

Token

Saldo



BNG

0 BNG

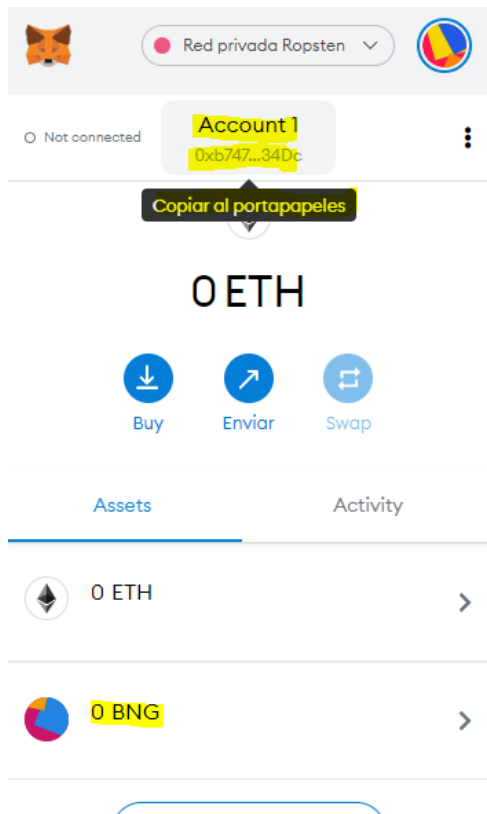
Atrás

Agregar tokens

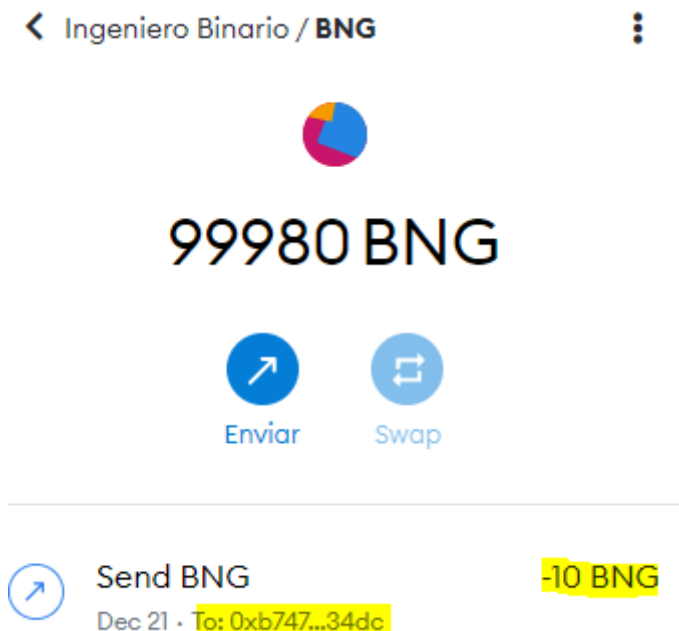




Pídele la dirección de la cartera, para poder enviarle monedas.

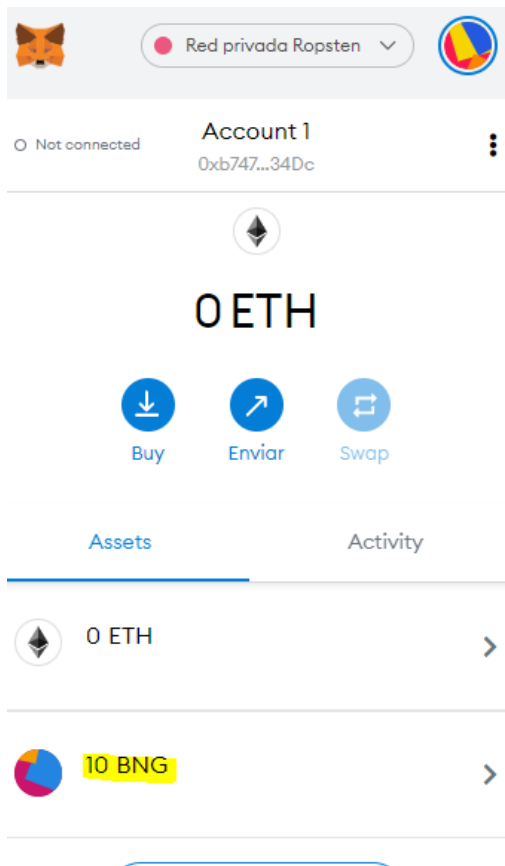


Una vez realices la transacción el usuario ya tendrá las monedas en su cartera.





Su cartera quedará así.





Espero que te haya gustado el documento explicativo para crear tu propia moneda.



¡Un saludo y cuídate!