2021-1-IE01-KA220-VET-000032943

Cofinanciado por la Unión Europea

TrainChain
Blockchain Training for Start Ups

Contenido básico de PR1-T3

Módulo 5 — Contratos inteligentes

Tutorial de programación

Autor del artículo: CCSDE

ID DEL PROYECTO:

Acuerdo de subvención	2021-1-IE01-KA220-VET-000032943
Programa	Erasmus+
Acción clave	KA220-VET — Asociaciones de cooperación en educación y formación profesionales
Campo	Educación y formación profesional
Acrónimo del proyecto	TrainChain
Título del proyecto	TrainChain — Blockchain Training for Start Ups
Fecha de inicio del proyecto	28/02/2022
Duración del proyecto	24 meses
Fecha de finalización del proyecto	27/02/2024

ser consideradas responsables del uso, que puede hacerse de la información contenida en las mismas.

HISTORIAL DE REVISIONES

Versión	Fecha	Autor	Descripción	Medidas de acción	Páginas
1.0	31/07/2022	CCSDE	Creación	C	8

(*) Acción: C = Creación, I = Insertar, U = Actualización, R = Reemplazar, D = Eliminar

DOCUMENTOS DE REFERENCIA

ID	Referencia		Título
1	2021-1-IE01-KA220-VET-000032943		Acuerdo de TrainChain
2			

DOCUMENTOS APLICABLES

ID	Referencia		Título
1			
2			

Contenido

1. Introducción

1.1 Descripción del módulo

Este módulo consta de 5 secciones principales para ayudar a los estudiantes a escribir e implementar su primer contrato inteligente. Los estudiantes comenzarán con la configuración de su entorno de desarrollo y luego pasarán a escribir su primer contrato inteligente simple. Configurarán su Metamask y usarán la red de prueba RSK para obtener algo de dinero digital de prueba e implementar su contrato. Finalmente, ejecutarán su contrato y evaluarán las acciones de su contrato en la cadena de bloques.

1.2 Objetivos del módulo

Los objetivos principales del módulo son que el alumno se familiarice con el trabajo con herramientas de desarrollo. El espacio criptográfico es un espacio tecnológicamente pesado e incluso los individuos enfocados en negocios necesitan tener una comprensión elemental de lo que implica la creación de soluciones blockchain.

1.3 Objetivos de aprendizaje

Los estudiantes que se a matriculen en este módulo echarán un vistazo bajo el capó de tecnologías muy complejas y desmitificarán la dificultad de interactuar con el emocionante espacio criptográfico.

1.4 Resultados de aprendizaje

Incluso los estudiantes que inician el módulo con cero conocimiento sobre cómo funciona la cadena de bloques, a lo largo de este tutorial conseguirán lo siguiente:

- Crear una billetera Metamask,
- Aprender los conceptos básicos de Solidity y Remix, el lenguaje de programación y la herramienta principal utilizada para crear contratos inteligentes,
- Aprender sobre la cadena de bloques RSK, su red principal y la red de prueba utilizada para implementar contratos inteligentes.

- Aprender qué es Gas y por qué se necesita para cambiar cosas en la cadena de bloques,
- Implementar un contrato inteligente real y ejecutarlo.

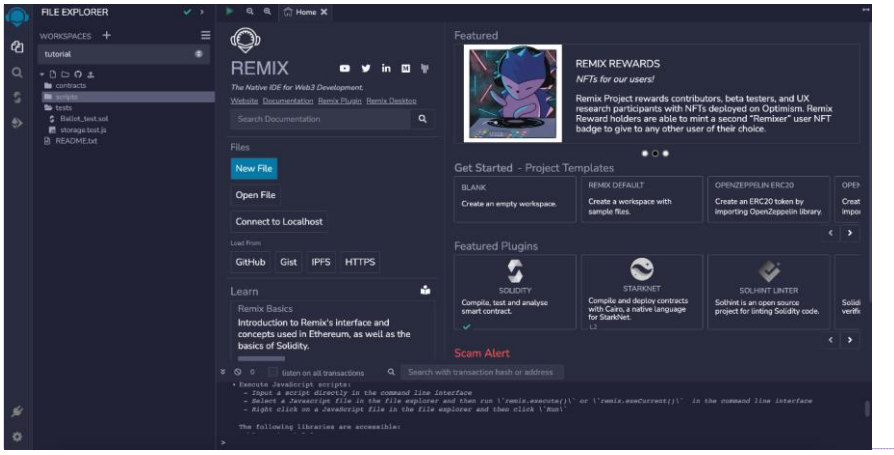
2. Contenido principal

2.1 Visión general

Solidity es el principal lenguaje de programación para escribir contratos inteligentes para la cadena de bloques Ethereum. Es un lenguaje orientado a contratos, lo que significa que los contratos inteligentes son responsables de almacenar toda la lógica de programación que realiza transacciones con la cadena de bloques. Es un lenguaje de programación de alto nivel que se parece mucho a JavaScript, Python y C++. Está diseñado para ejecutarse en la máquina virtual Ethereum (EVM), que está alojada en nodos Ethereum conectados a la cadena de bloques. Se escribe estáticamente, y apoya la herencia, las bibliotecas y más. En resumen, tiene toda la capacidad que necesita para construir aplicaciones de blockchain de fuerza industrial.

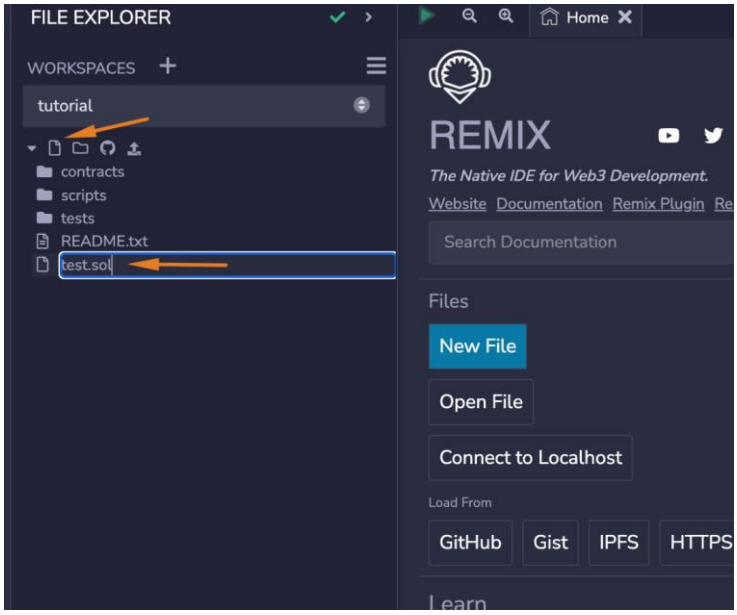
2.2 Configuración del entorno REMIX IDE

Vamos a usar Remix para escribir todo el código en este tutorial. Remix es un IDE basado en un navegador que permite escribir, compilar e implementar contratos inteligentes con excelentes características como el almacenamiento de archivos persistente. Usaremos Remix para que no tener que descargar ninguna herramienta de desarrollador o instalar nada para comenzar. Dirígete a Remix Online IDE para seguir este tutorial.



Commented [Ma1]: Considerar la legibilidad de las cifras, Considere la posibilidad de dar a las figuras un encabezamiento, por ejemplo, la figura 1, etc.... y hacer referencia a la figura en el texto.

Crear un nuevo archivo con el nombre test.sol (sol es la extensión para el archivo Solidity)



2.3 Escribiendo nuestro primer contrato inteligente

Comencemos a escribir lo que se conoce como nuestro pragma. El pragma se requiere al principio de todos los archivos de Solidity y lo que esto hace es decirle a Solidity qué versión del compilador necesita usar este archivo.

Después de hacer nuestra línea pragma, lo siguiente que tenemos que hacer es definir un contrato.

```
1   pragma solidity 0.8.10;
2
3   contract SimpleStorage {
4       uint storedData;
5
6       function set(uint x) public {
7           storedData = x;
8       }
9
10      function get() public view returns (uint) {
11          return storedData;
12      }
13  }
```

Este contrato inteligente tiene:

- Una variable de datos almacenados para almacenar un número
- Una función get() para devolver el número almacenado en la variable Datos almacenados
- Un función set() para cambiar el número almacenado en la variable Datos almacenados

Ahora que tenemos nuestro primer contrato, lo que tenemos que hacer es compilarlo y luego implementarlo. Todos nuestros contratos en Solidity deben compilarse en bytecode. Este bytecode se envía a la red Ethereum donde se implementa el contrato.

Después de la implementación del contrato, comienza a vivir en la cadena de bloques y podemos llamarlo en cualquier momento.

Antes de empezar a pensar en implementar nuestro contrato, primero debemos conectar el Remix IDE con MetaMask.

2.4 Configuración de MetaMask

MetaMask es una especie de monedero web que facilita las transacciones utilizando sus cuentas. También se puede utilizar con redes RSK. Tiene versiones para varios navegadores, como Chrome, Firefox, Opera y Brave.

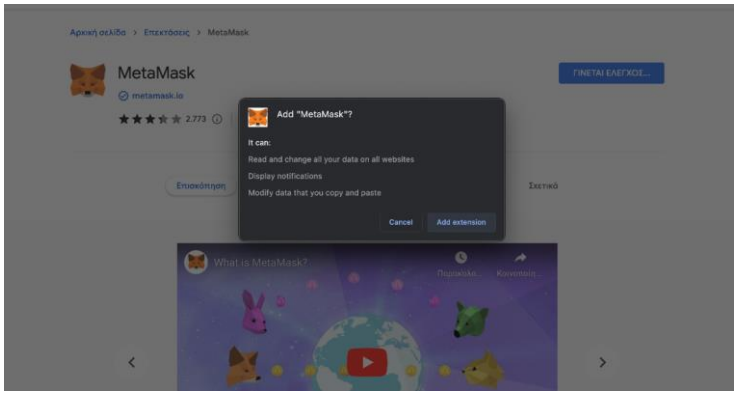
1. Ve a **metamask.io** e instálalo.

Puede utilizar la herramienta metamask **-landing.rifos.org para** descargar/instalar Metamask, y agregar la cadena de **bloques de contrato inteligente RSK** o seguir los pasos enumerados en **metamask.io**.

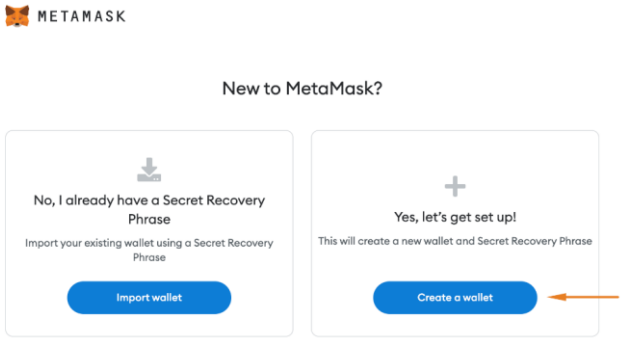
2. Crea una cuenta.

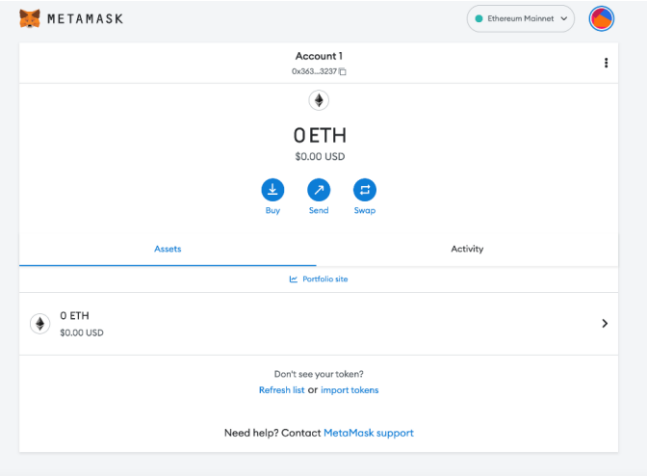
Anota tu *seed phrase* de 12 palabras, o mnemónica, o frase de respaldo (todos estos términos significan lo mismo). Esto se utiliza para recuperar tus cuenta, en caso de que pierdas tu contraseña. No hay otra manera de recuperarlo. ¡El objetivo de las tecnologías blockchain es que su naturaleza descentralizada y los datos cifrados te dan el control total de tus datos!

¡La *seed phrase* es lo más importante en una billetera/cuenta!



Creación de una cuenta (Wallet) en MetaMask



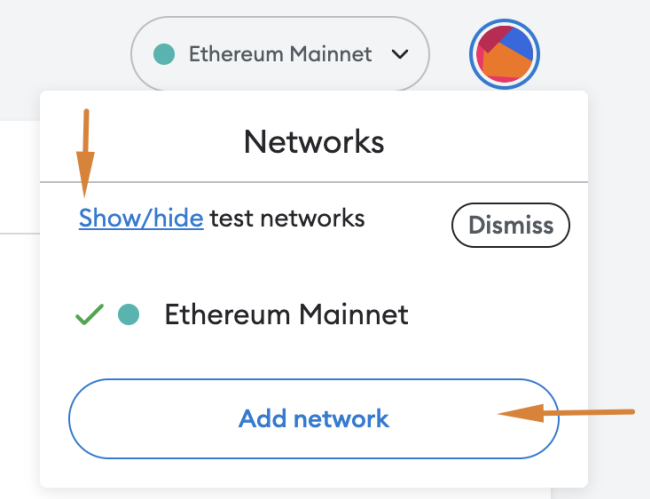


Remix es una herramienta web en línea. Es un IDE (Integrated Development Environment) utilizado para escribir, compilar, implementar y depurar código de Solidity. Se puede conectar con Metamask y luego se utiliza para implementar contratos inteligentes tanto en RSK **Testnet como en Mainnet.**

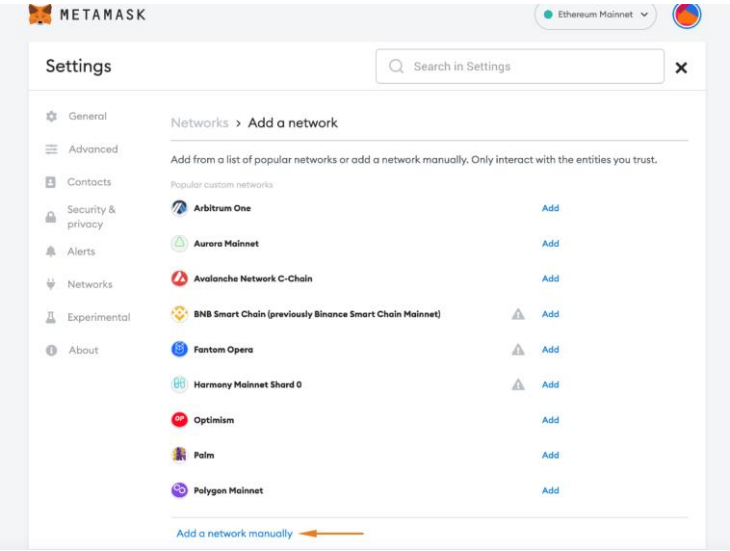
Haga clic aquí para ir al **IDE de Remix en línea.**

3. Conecta MetaMask a la RSK Testnet

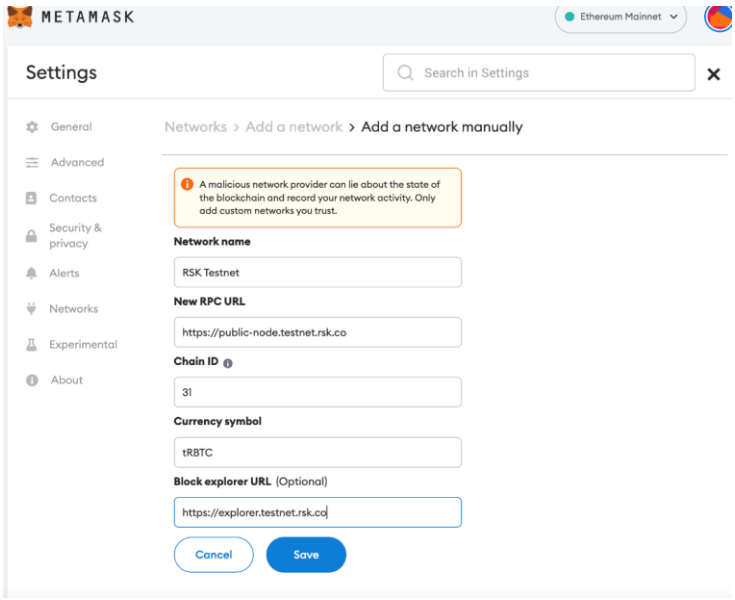
- Ir a las redes



- Agregar red manualmente



- RPC personalizado completo



- Después de configurarlo, selecciona la RSK Testnet.

Grifo de Testnet

Puede obtener un poco de Testnet RBTC en faucet.testnet.rsk.co.

¿Qué es RBTC?

RBTC es dinero digital, utilizado como gas para pagar la ejecución de contratos inteligentes en la red, como la tarifa de transacción para el comercio de tokens del ecosistema RSK, de la misma manera que ETH se utiliza como gas para Ethereum.

¿Qué es una tarifa de gas?

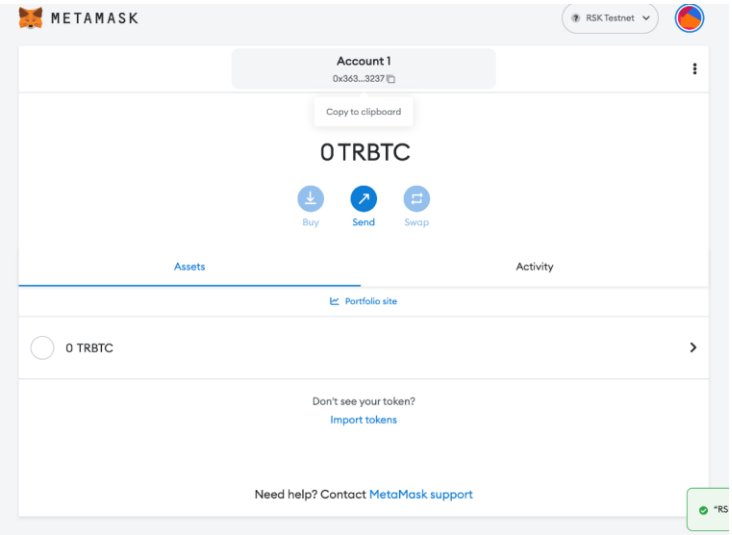
Una tarifa de gas es una tarifa de transacción de blockchain, pagada a los validadores de red por sus servicios a la cadena de bloques. Sin las tarifas, no habría ningún incentivo para que nadie aposte su ETH y ayude a asegurar la red.

¿Por qué tengo que pagar una tarifa de gas?

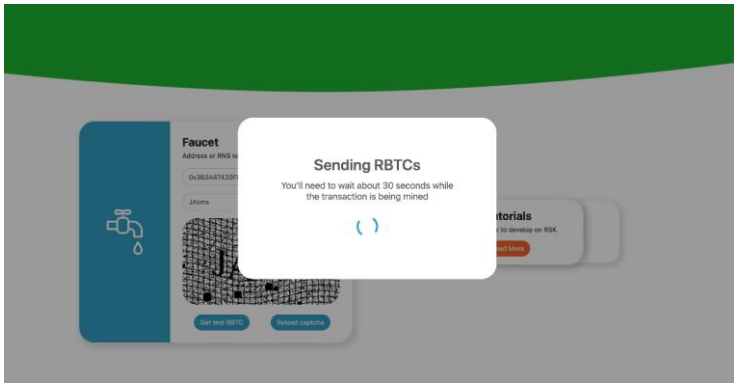
La tarifa de gas existe para pagar a los validadores de la red por su trabajo asegurando la cadena de bloques y la red. Sin los honorarios, habría pocas razones para apostar ETH

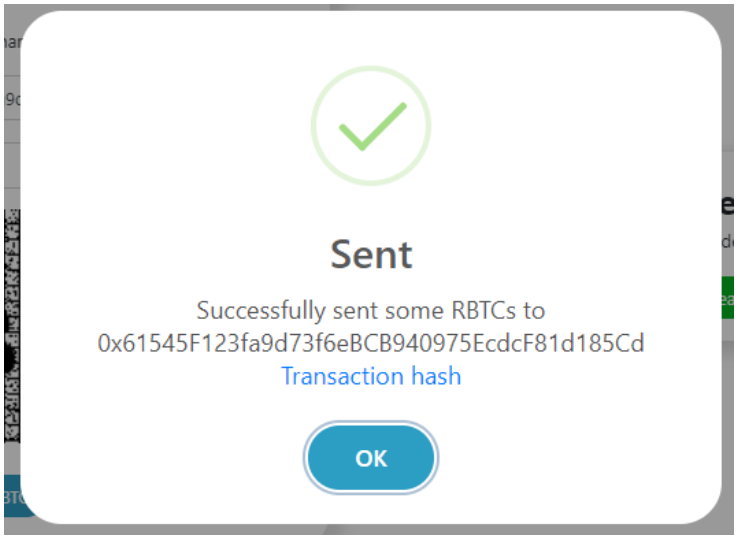
y convertirse en un validador. La red estaría en riesgo sin validadores y sin el trabajo que realizan.

- Copia tu dirección de Metamask,



- Ingresa la dirección de tu billetera, pasa el CAPTCHA y espera unos segundos...





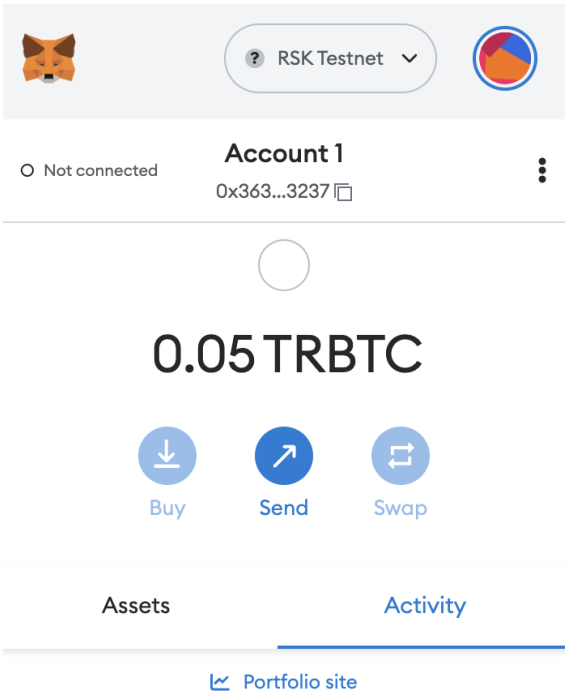
Puedes ver el hash de la transacción en la cadena de bloques.

Por ejemplo, en nuestro ejemplo: 0x47b93cdce6fe4473a20867c5cf9bf74195db14cb74a3aa9a4e8b7908fd367167.

¿Qué es un hash de transacción?

Un hash/ID de transacción (a menudo abreviado como hash tx o hash txn) es **un identificador único, similar a un recibo, que sirve como prueba de que una transacción fue validada y añadida a la cadena de bloques**. En muchos casos, se necesita un hash de transacción para localizar fondos.

¡Ahora tienes un poco de RBTC!



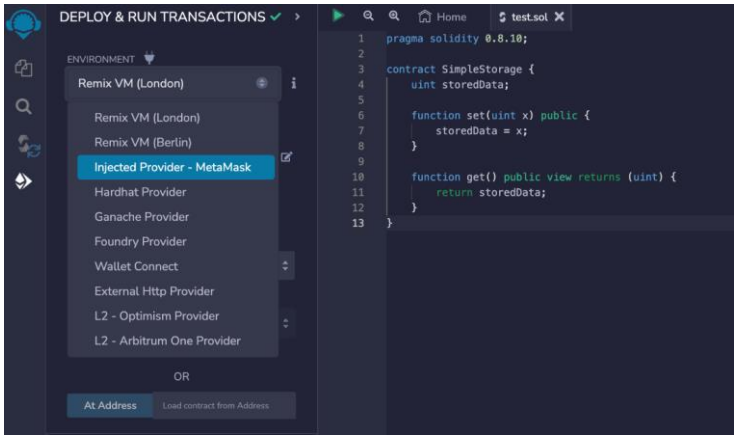
Ahora que hemos completado nuestra configuración de MetaMask podemos volver a Remix para implementar nuestro contrato

2.5 Conecte Remix a la RSK TESTNET

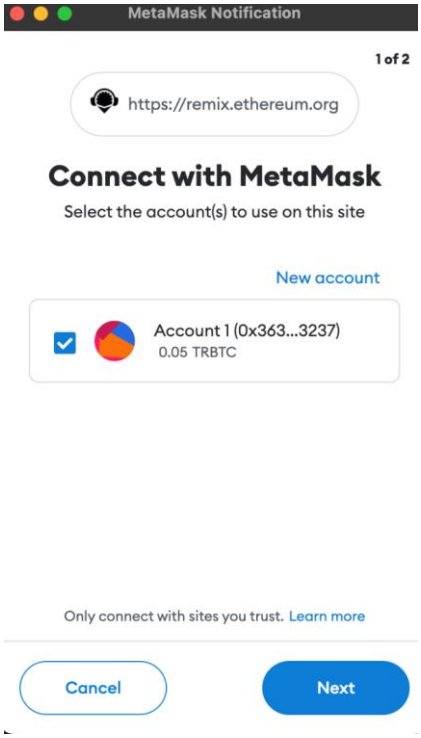
Con la red RSK seleccionada en Metamask...

En Remix, en el lado izquierdo, localiza el botón Deploy & run transactions.

En Environment, elije Injected Provider — MetaMask



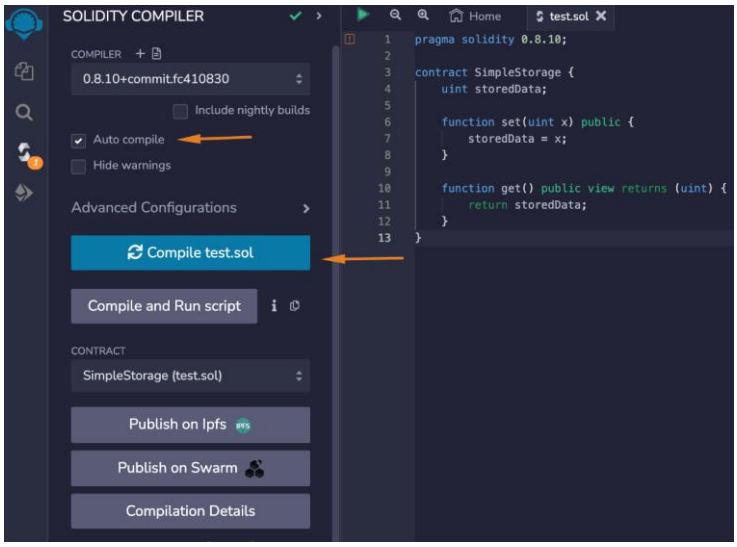
Injected Provider — MetaMask conecta Remix con tu cuenta activa en Metamask.



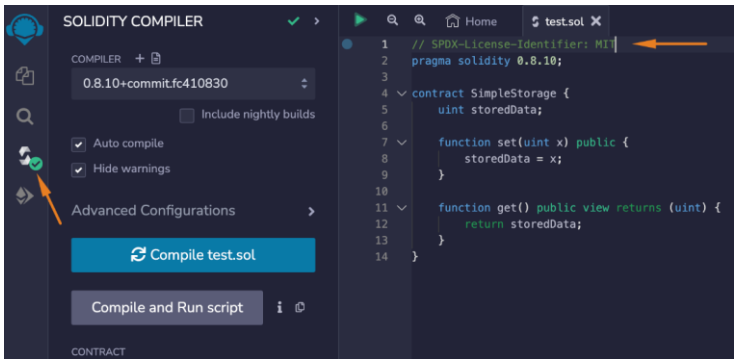
2.6 Compila tu contrato inteligente

Encuentra el tercer botón en el lado izquierdo y haz clic en el compilador Solidity (¡asegúrate de habilitar la autocompilación, te ahorrará mucho tiempo!)

Haz clic en el botón Compile test.sol



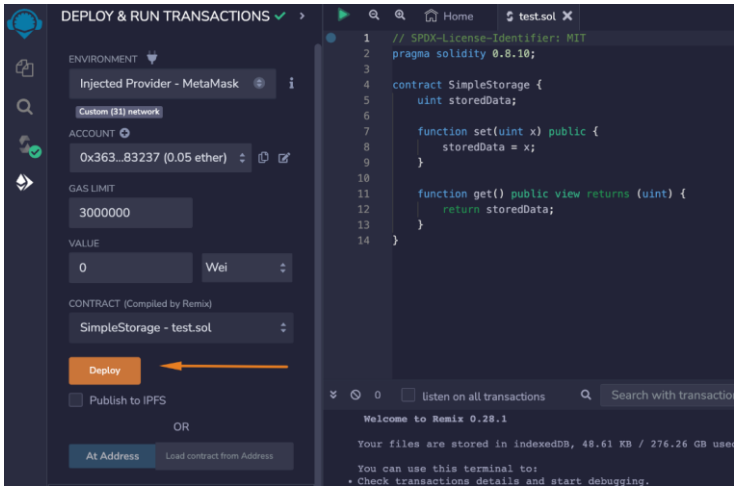
Comprueba el signo verde que te dará el mensaje de compilación exitosa.



El consejo: Para evitar advertencias añade comentario a la línea 1: //SPDX-license-Identificador: MIT

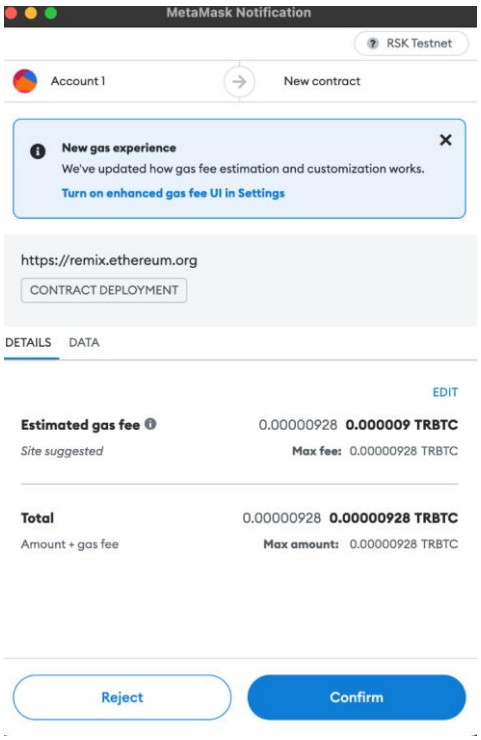
2.7 Implementar un contrato inteligente en RSK TESTNET

En el panel izquierdo, ve al botón Deploy & run transactions.



Por ahora solo tenemos un contrato inteligente, por lo que se selecciona automáticamente en el menú desplegable.

Se abrirá una ventana emergente Metamask, para confirmar la transacción y crear la prueba de contrato inteligente.sol



Haz clic en Confirmar.

En la parte inferior derecha, tendremos el mensaje de creación de SimpleStorage pendiente...

Una vez confirmado, podemos echarle un vistazo más de cerca.

Haz clic en la línea de transacción o en el botón de debug (en el lado derecho) para ver más detalles de la transacción.

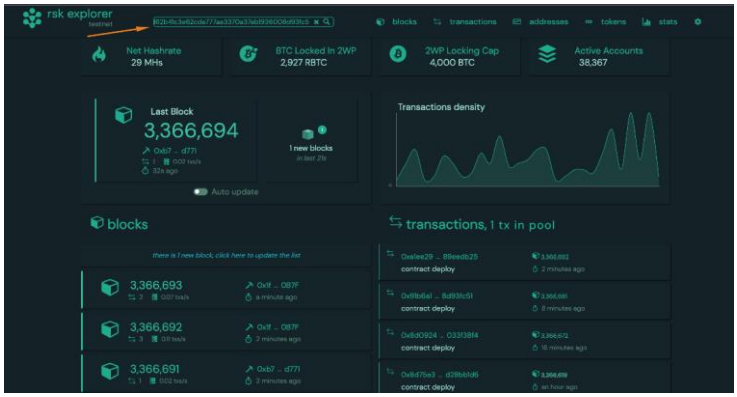
Copie el hash de la transacción para verificarlo usando el explorador de cadenas de bloques RSK

Es este ejemplo, el hash de la transacción es:

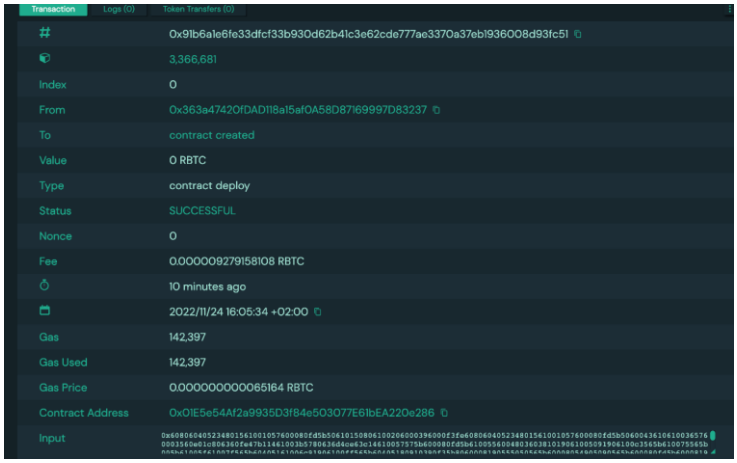
0x91b6a1e6fe33dfcf33b930d62b41c3e62cde777ae3370a37eb1936008d93fc51

2.8 Explorador de RSK

El explorador RSK es el explorador de blockchain para las transacciones RSK. Utilizaremos el **explorador Testnet**



Esto es lo que esperaríamos ver:

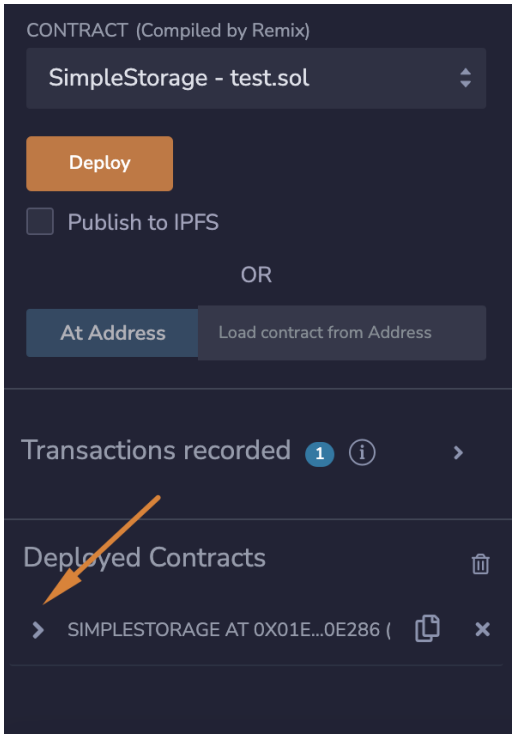


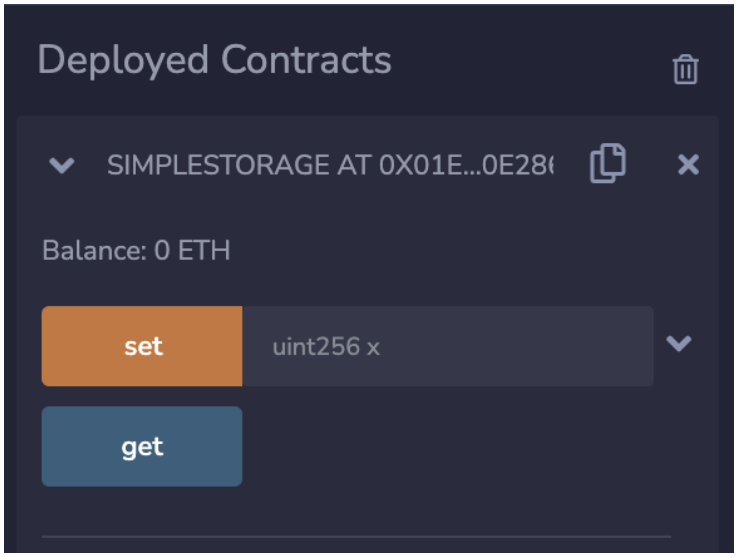
Puede verificar el hash de ejemplo de este tutorial en:

0x91b6a1e6fe33dfcf33b930d62b41c3e62cde777ae3370a37eb1936008d93fc51

2.9 Interactúa con tu contrato inteligente

Cuando se implementa un contrato inteligente con Remix, podemos verlo en el panel izquierdo bajo implementar y ejecutar transacciones:


CONTRACT (Compiled by Remix)
SimpleStorage - test.sol
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded 1
Deployed Contracts
SIMPLESTORAGE AT 0X01E...0E286 (



¡Estas son las mismas funciones que creamos en nuestro contrato inteligente!

Los botones naranjas son funciones que cambiarán cierta información almacenada en la cadena de bloques.

Cada vez que lo llamamos, la función gastará algo de gas y realizará lo que estaba programado para hacer.

Los botones azules son funciones que son de solo lectura y no cambian nada almacenado en la cadena de bloques, solo obtienen datos. No gastamos gas al usarlos, solo la función que cambia las cosas en la cadena de bloques tiene un costo.

Obtén el valor de la Blockchain

En primer lugar, comprobaremos el valor almacenado en el momento de la implementación.

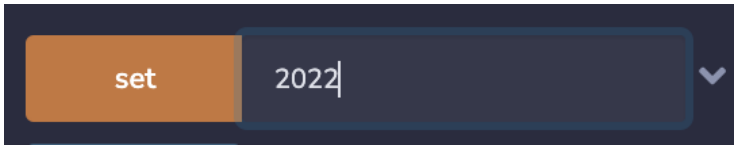
Haz clic en el botón get

No tenemos ningún valor almacenado, porque aún no hemos definido nada.

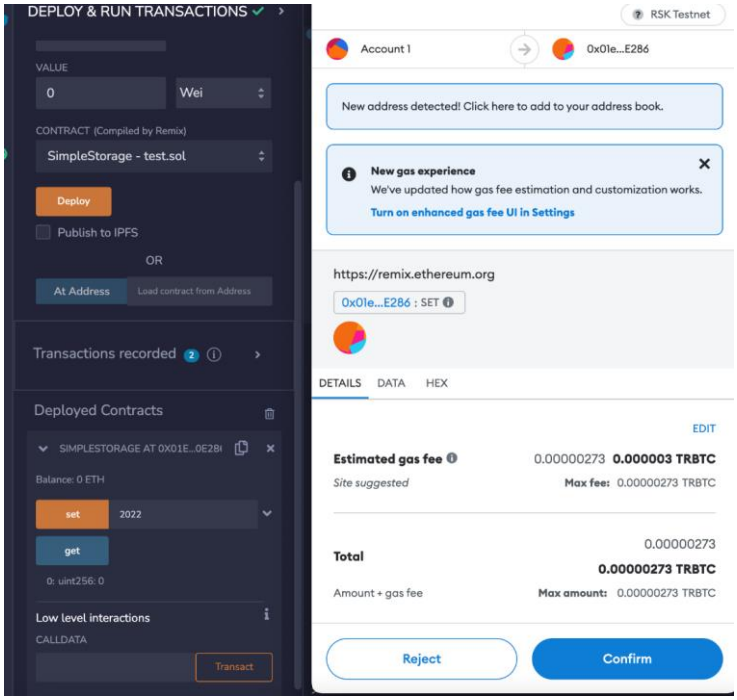
En la parte inferior derecha, podemos ver que se hizo una llamada a la función SimpleStorage.get.

Establecer el valor en la Blockchain

Pon un valor en el campo en el lado derecho del botón de configuración y haz clic en el botón



Se abrirá una ventana emergente de Metamask, para confirmar la transacción para almacenar el valor en la Blockchain. La confirmación es necesaria porque como se describió anteriormente, ahora haremos un cambio en la cadena de bloques y tendremos que gastar algo de gas para hacerlo.

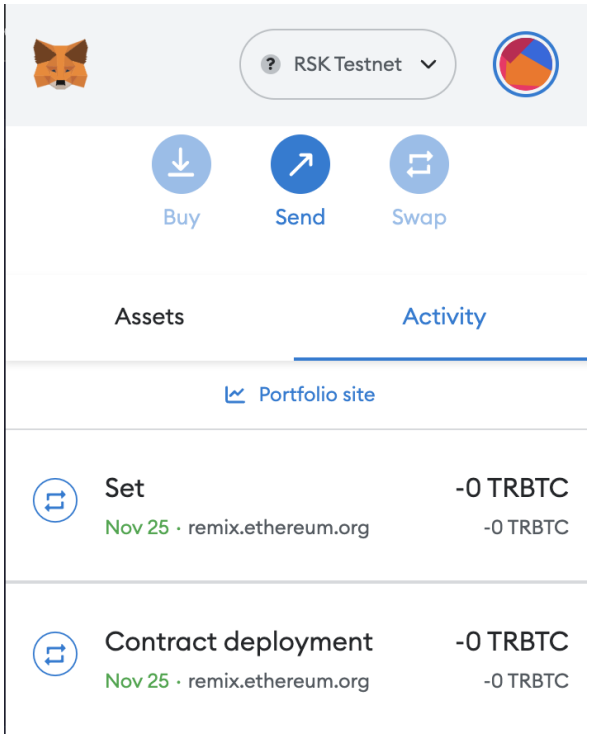


Haz clic en Confirm

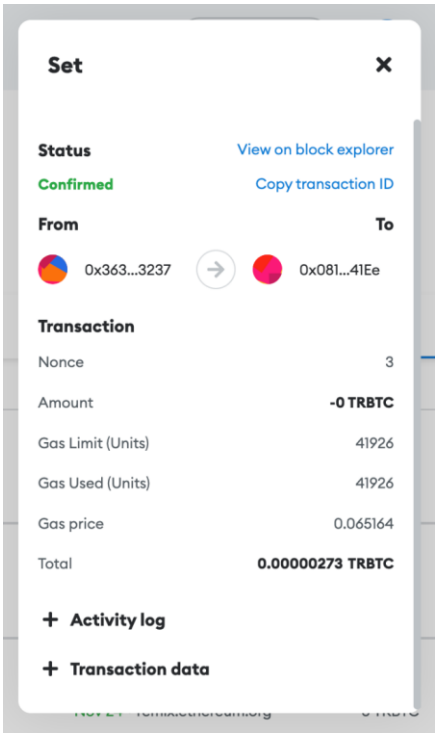
En la parte inferior derecha, podemos verificar que la transacción está pendiente, a la espera de confirmación en blockchain:

```
transact to SimpleStorage.set pending ...
```

¡Después de unos segundos, Metamask mostrará cuándo se ha confirmado la transacción!



Haz clic en Establecer para ver los detalles de la transacción

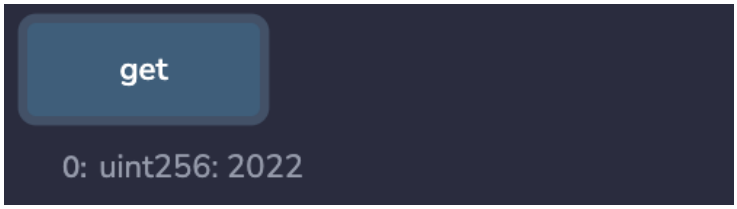


Puedes verificarlo en el explorador RSK al igual que lo hicimos antes.

Obtener (de nuevo)

Ahora que tenemos el valor 2022 guardado, podemos volver a preguntar desde la cadena de bloques.

Haz clic en el botón get una vez más.



¡El valor devuelto es correcto!

¿A dónde voy desde aquí?

Todo depende de tu negocio y tu imaginación. Los contratos inteligentes se pueden utilizar para registrar y asegurar transacciones financieras, pueden usarse para representar artículos vendibles en juegos, asegurar documentos y decisiones legales o servir como escrituras digitales en el espacio inmobiliario. El cielo es el límite.