



PR1-T3 Core Content

Module 5 – Smart Contracts

Hands-on programming tutorial

Author: CCSDE

PROJECT ID:

Grant agreement	2021-1-IE01-KA220-VET-000032943
Programme	Erasmus+
Key action	KA220-VET - Cooperation partnerships in vocational education and training
Field	Vocational Education and Training
Project acronym	TrainChain
Project title	TrainChain - Blockchain Training for Start Ups
Project starting date	28/02/2022
Project duration	24 months
Project end date	27/02/2024

Disclaimer: This project is funded with the support of the European Commission. The information and views set out in this document are those of the author(s) and do not necessarily reflect the official opinion of the European Commission. Neither the European Union institutions nor any person acting on their behalf may be held responsible for the use, which may be made of the information contained therein.

REVISION HISTORY

Version	Date	Author	Description	Action	Pages
1.0	31/07/2022	CCSDE	Creation	C	8
1.1	31/12/2022	CCSDE	Update	U	30

(*) Action: C = Creation, I = Insert, U = Update, R = Replace, D = Delete

REFERENCED DOCUMENTS

ID	Reference		Title
1	2021-1-IE01-KA220-VET-000032943		TrainChain Agreement
2			

APPLICABLE DOCUMENTS

ID	Reference		Title
1			
2			

Contents

1. Introduction	5
1.1 Module Description	5
1.2 Module Goals	5
1.3 Learning Objectives	5
1.4 Learning Outcomes	5
2. Main Content	6
2.1 Overview-Solidity	6
2.2 Environment setup REMIX IDE	7
2.3 Writing our first Smart Contract	8
2.4 Metamask Configuration.....	9
2.5 Connect Remix to the RSK TESTNET	16
2.6 Compile your Smart Contract	18
2.7 Deploy a Smart Contract on the RSK TESTNET	19
2.8 RSK Explorer.....	21
2.9 Interact with your Smart Contract	23
3. Knowledge Assessment	29
4. References.....	30

1. Introduction

1.1 Module Description

This module consists of 5 main sections to help the learners write and deploy their first smart contract. Learners will start with setting up their dev environment, then will move on to write their first simple smart contract. They will setup their Metamask and use the RSK testnet to get some test digital money and deploy their contract. Finally, they will run their contract and assess the actions of their contract on the blockchain.

1.2 Module Goals

The module's main goals are for the learner to gain familiarity with working with dev tools. The crypto space is a tech heavy space and even business focused individuals need to have an elementary understanding of what goes into creating blockchain solutions.

1.3 Learning Objectives

Learners that will apply themselves through this module will take a look under the hood of very complex technologies and demystify the difficulty of interacting with the exciting crypto space.

1.4 Learning Outcomes

Even learners that start the module with zero knowledge about how the blockchain works will, throughout this tutorial,

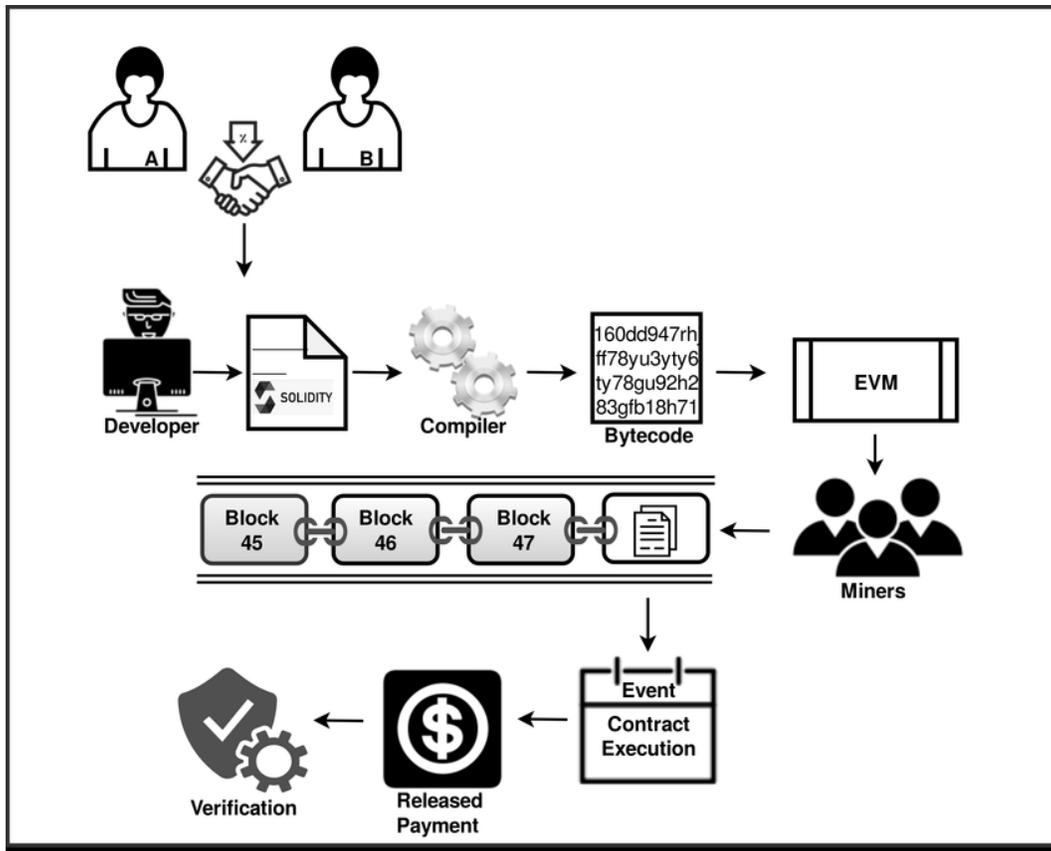
- Created a Metamask wallet,
- Learn the 101 basics of Solidity and Remix, the programming language and the main tool used to create smart contracts,
- Learn about the RSK blockchain, its main network and the test network used to deploy smart contracts,
- Learn what Gas is and why it is required to change things on the blockchain,
- Deploy an actual smart contract and run it.

2. Main Content

The 'Ingredients'

Before moving to programming we need first to clarify both the task itself and the tools we need to use to accomplish the task (what's their name, their role etc).

So let's see first how a smart contract is created and applied and recall some knowledge from previous modules:



Total cycle of smart contract execution over Ethereum blockchain.
From Researchgate: available via license: Creative Commons Attribution 4.0 International

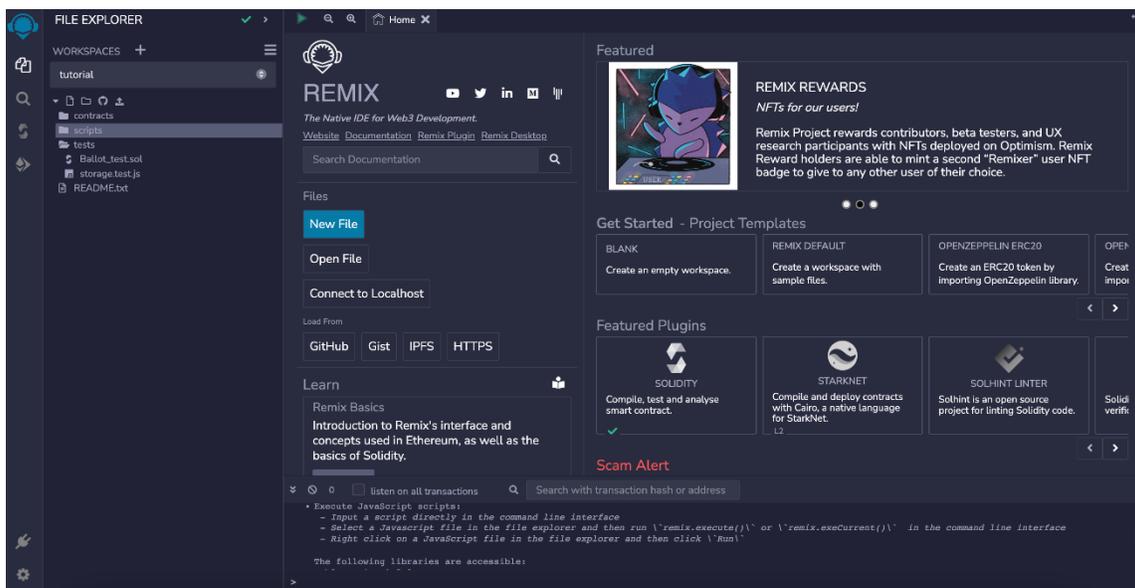
2.1 Overview-Solidity

Solidity is the main programming language for writing smart contracts for the Ethereum blockchain. It is a contract-oriented language, which means that smart contracts are responsible for storing all of the programming logic that transacts with the blockchain. It's a high-level programming language that looks a lot like JavaScript, Python, and C++. It's designed to run on the Ethereum Virtual Machine (EVM), which is hosted on Ethereum Nodes that are connected to the blockchain. It is statically typed, and

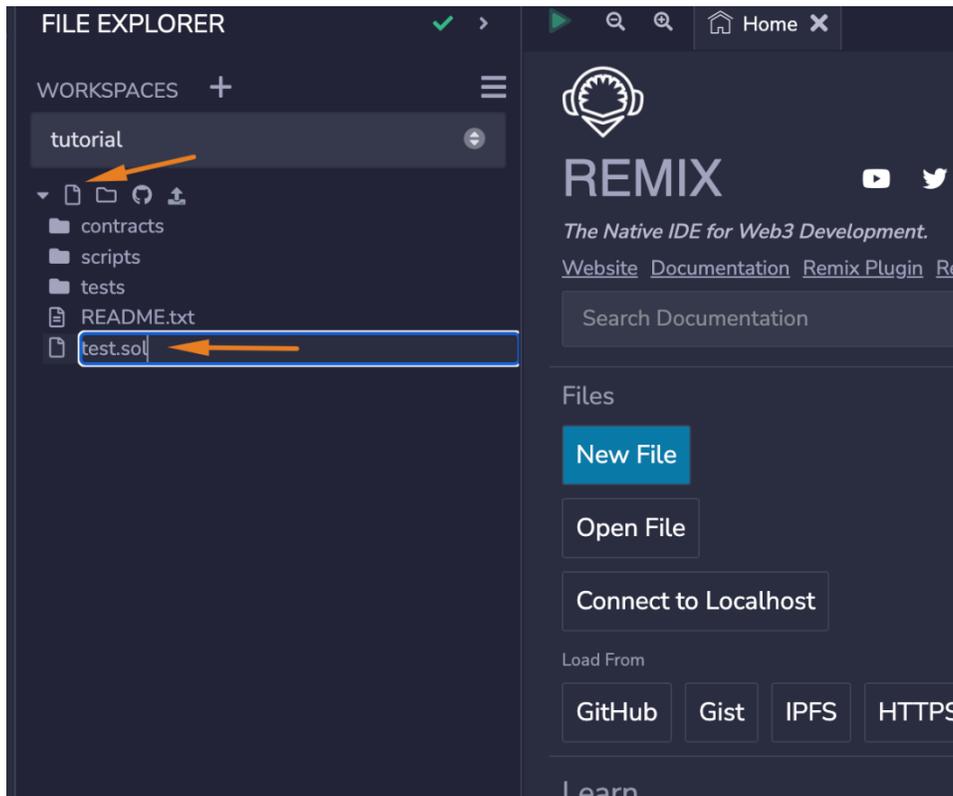
supports inheritance, libraries, and more! In short, it has all the capability that you need in order to build industrial strength blockchain applications.

2.2 Environment setup REMIX IDE

We're going to use Remix to write all of the code in this tutorial. Remix is a browser-based IDE that allows you to write, compile and deploy smart contracts boasting excellent features like persistent file storage. We'll use Remix so that we don't have to download any developer tools or install anything to get started. Head on over to the Remix Online IDE to follow along with this tutorial.



Create a new file with name test.sol (sol is the extension for Solidity file)



2.3 Writing our first Smart Contract

And let's start by writing what's known as our pragma. The pragma is required at the beginning of all of your Solidity files and what this does is actually tell Solidity what compiler version this file needs to use.

After we do our pragma line, the next thing we need to do is to define a contract.

```
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 }
```

This smart contract has:

- A variable stored Data to store a number
- A function get() to return the number stored at variable stored Data
- A function set() to change the number stored at variable stored Data

Now that we have our first contract, what we need to do is to compile and then deploy it. All of our contracts in Solidity need to be compiled into bytecode. This bytecode is then sent to the Ethereum network where the contract is deployed.

After the contract deployment, it starts living on the blockchain and we can call it anytime.

Before we even begin thinking about deploying our contract, we first must connect the Remix IDE with MetaMask.

2.4 Metamask Configuration

Metamask is a kind of web wallet which facilitates transactions using yours accounts. It can be used with RSK networks too. It has versions for several browsers, like Chrome, Firefox, Opera and Brave.

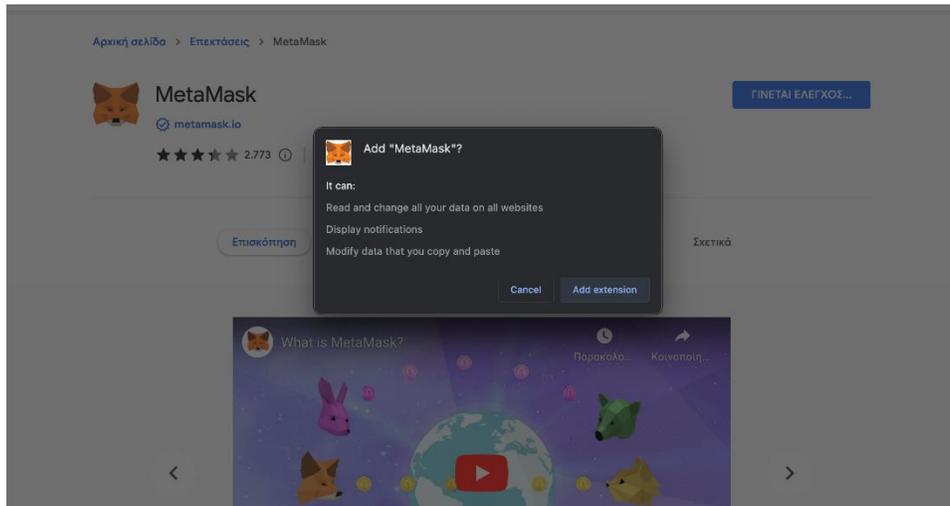
1. Go to metamask.io and install it.

You can either use the metamask-landing.rifos.org tool to download/install Metamask, and add the [RSK smart contract blockchain](https://rsk.smartcontractblockchain.com) or follow the steps listed in metamask.io.

2. Create an account.

Write down your 12 word seed phrase, or mnemonic, or backup phrase (all these terms mean the same). This is used to recover your account, in case you lose your password. There is NO other way to recover it. The whole point of blockchain technologies is that their decentralized nature and encrypted data give the total control of your data to you!

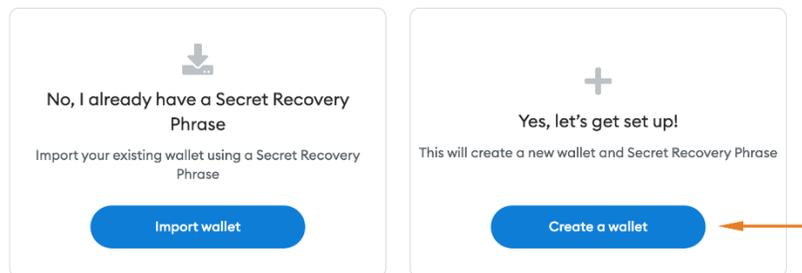
The seed phrase is the most important thing in a wallet/account!

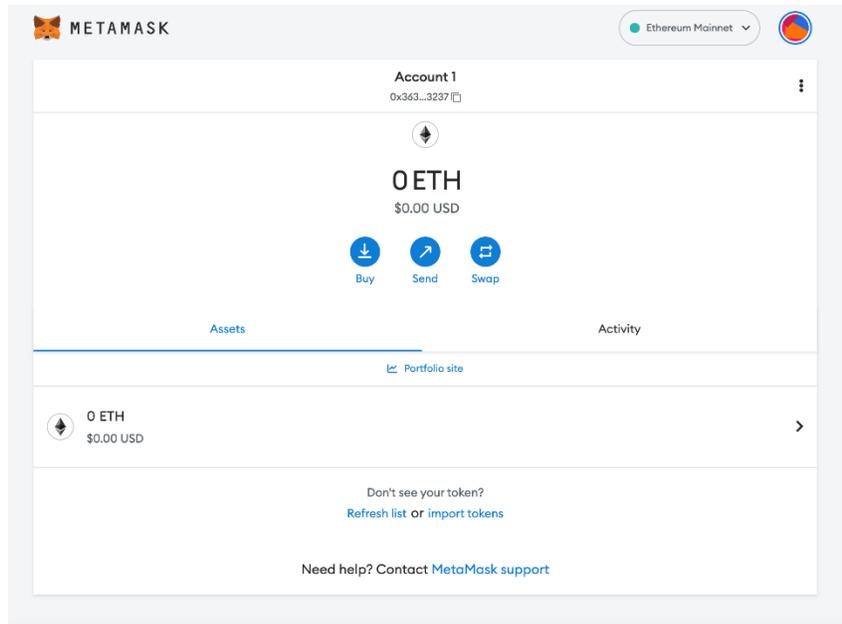


Creating an Account (Wallet) in MetaMask



New to MetaMask?

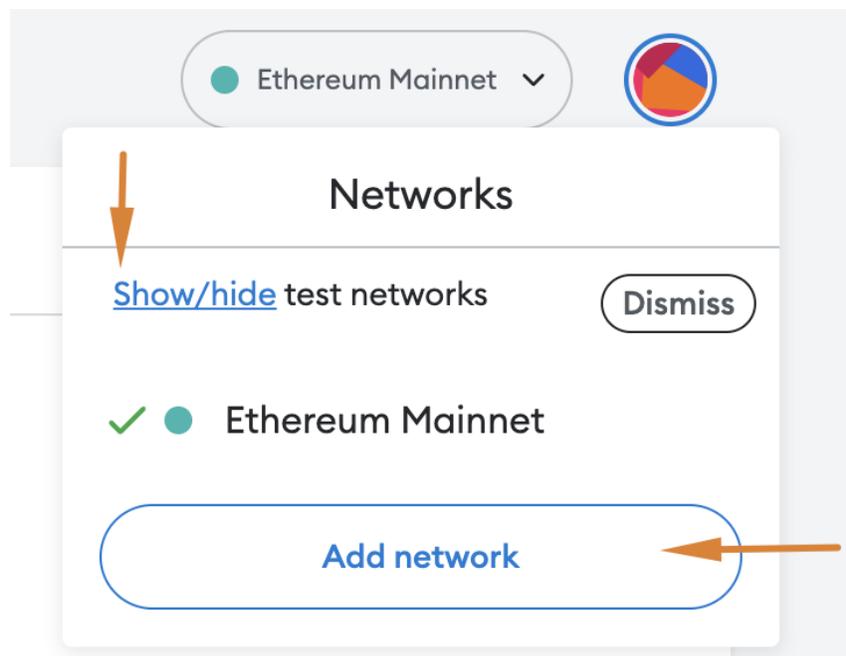




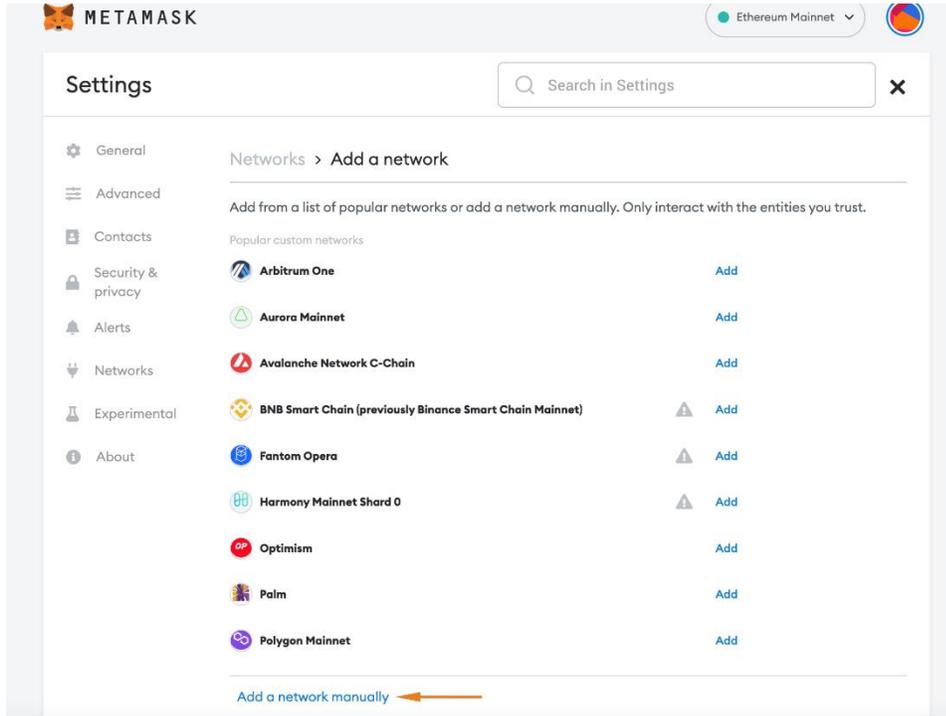
Remix is an online web tool. It is an IDE (Integrated Development Environment) used to write, compile, deploy and debug Solidity code. It can be connected with Metamask and then used to deploy smart contracts to both the [RSK Testnet and Mainnet](#)

Click here to go to the [online Remix IDE](#).

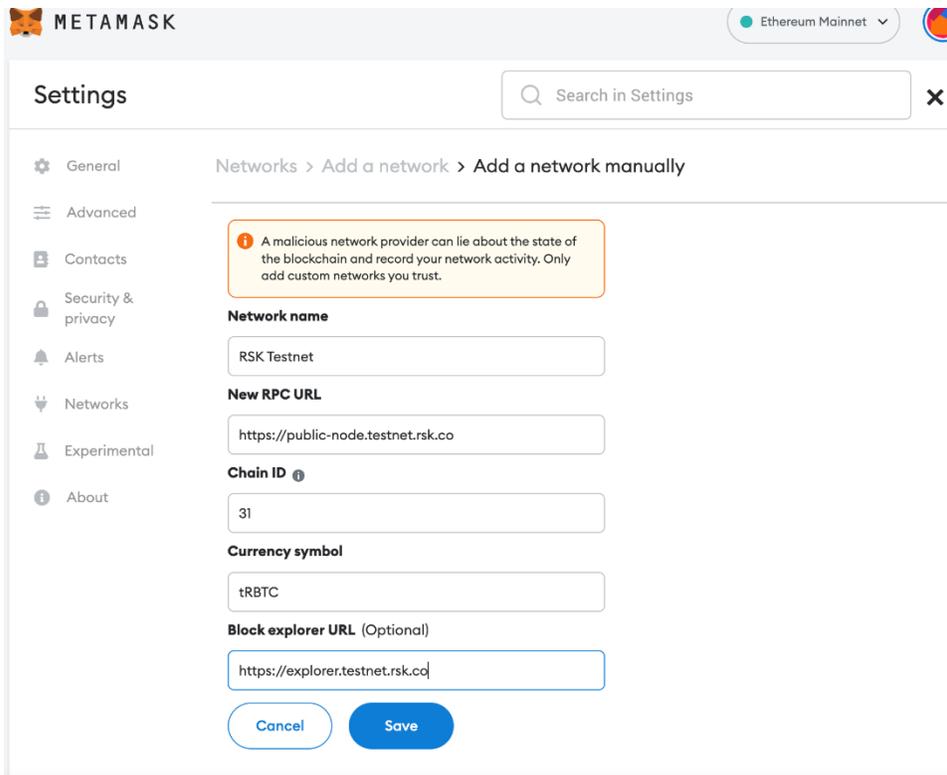
3. Connect MetaMask to the RSK Testnet
 - Go to networks



- Add network manually



- Complete Custom RPC



- After configuring it, select the RSK Testnet.

Testnet Faucet

You can get some Testnet RBTC at faucet.testnet.rsk.co.

What is RBTC?

RBTC is digital money, used as gas to pay for smart contract execution on the network, such as the transaction fee for trading RSK ecosystem tokens, the same way as ETH is used as gas for Ethereum.

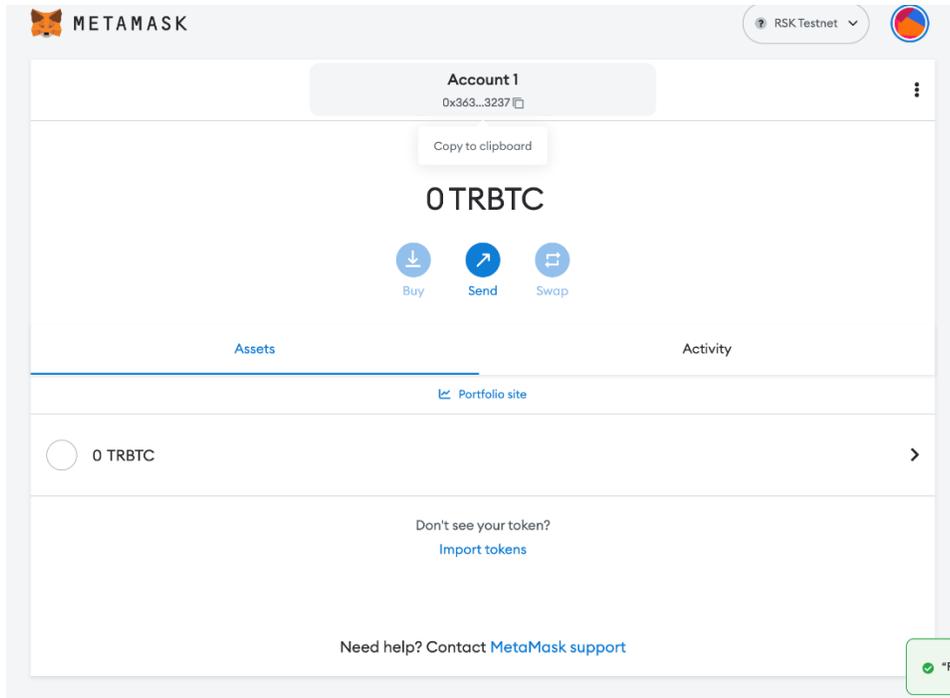
What Is a Gas Fee?

A gas fee is a blockchain transaction fee, paid to network validators for their services to the blockchain. Without the fees, there would be no incentive for anyone to stake their ETH and help secure the network.

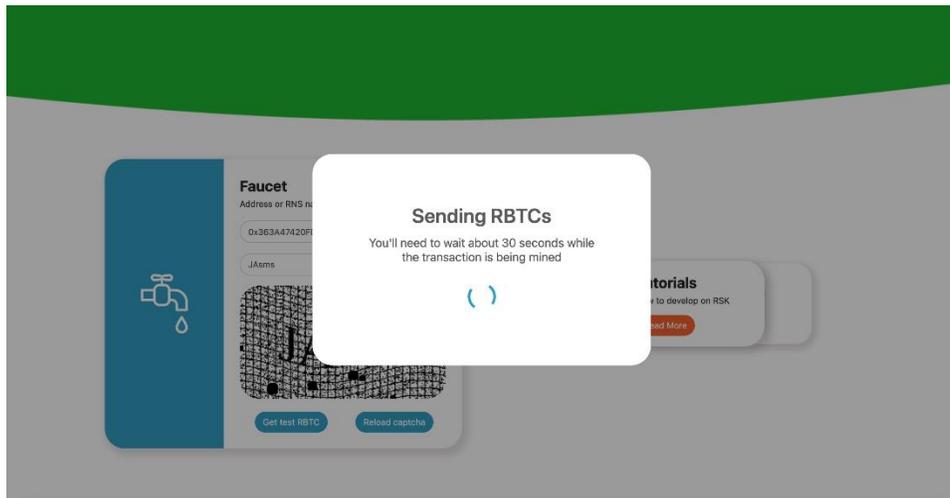
Why Do I Have to Pay a Gas Fee?

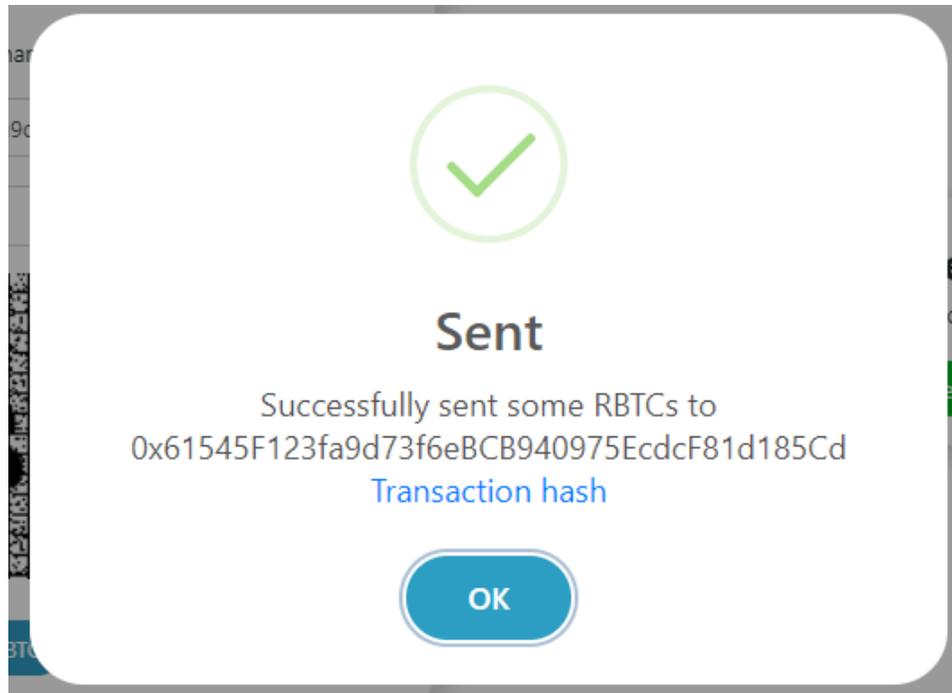
The gas fee exists to pay network validators for their work securing the blockchain and network. Without the fees, there would be few reasons to stake ETH and become a validator. The network would be at risk without validators and the work they do. Wi

- Copy your address from Metamask,



- Enter your wallet address, pass the CAPTCHA and wait a few seconds...





You can see the transaction hash on the blockchain.

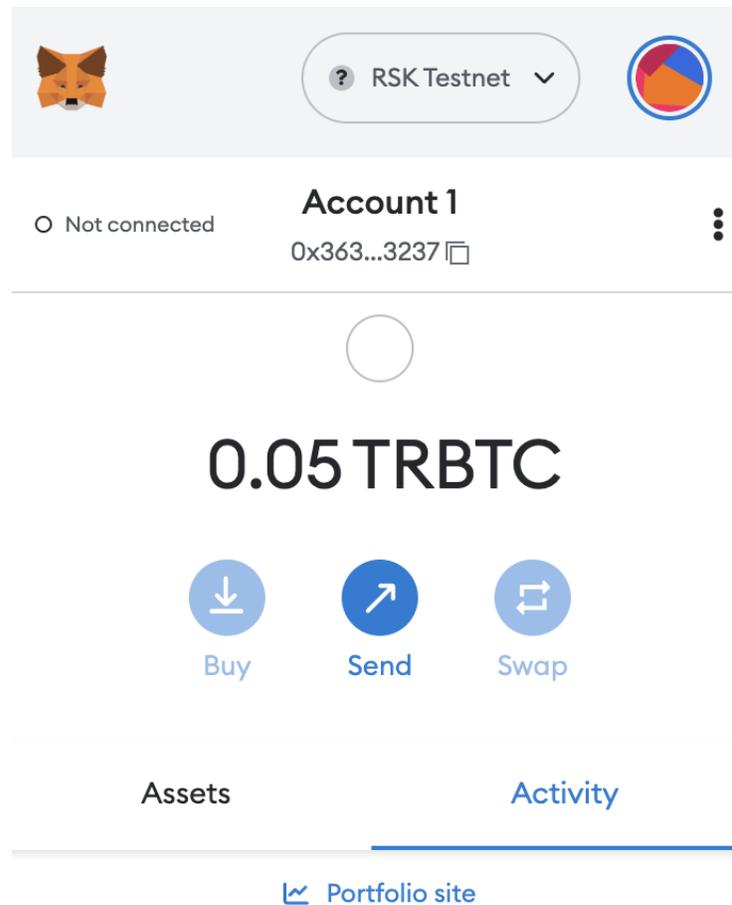
For example on our example ->

[0x47b93cdce6fe4473a20867c5cf9bf74195db14cb74a3aa9a4e8b7908fd367167](https://explorer.btc.com/tx/0x47b93cdce6fe4473a20867c5cf9bf74195db14cb74a3aa9a4e8b7908fd367167).

What is a transaction hash?

A transaction hash/ID (often abbreviated as tx hash or txn hash) is **a unique identifier, similar to a receipt that serves as proof that a transaction was validated and added to the blockchain.** In many cases, a transaction hash is needed in order to locate funds.

Now you have some RBTC!



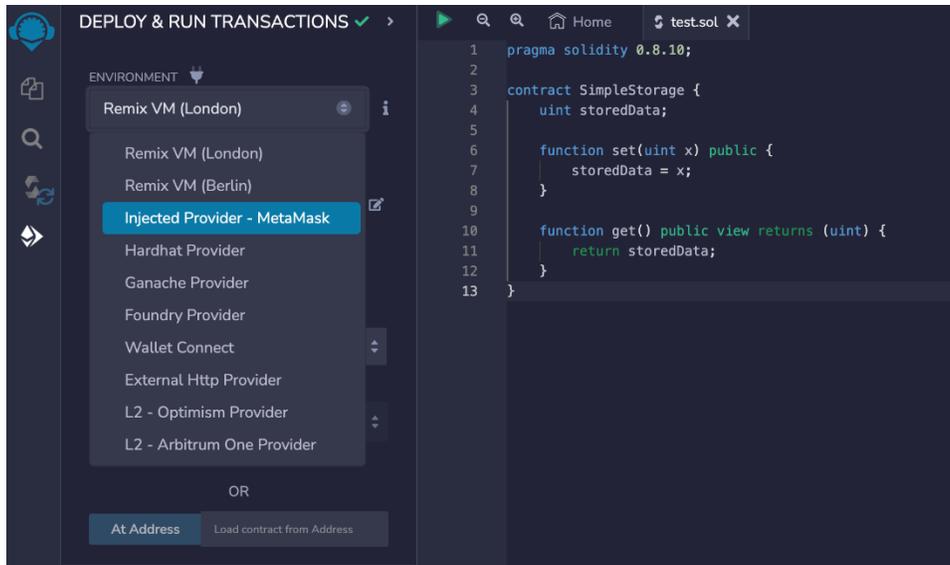
Now that we have completed our MetaMask Configuration we can return to Remix to Deploy our Contract

2.5 Connect Remix to the RSK TESTNET

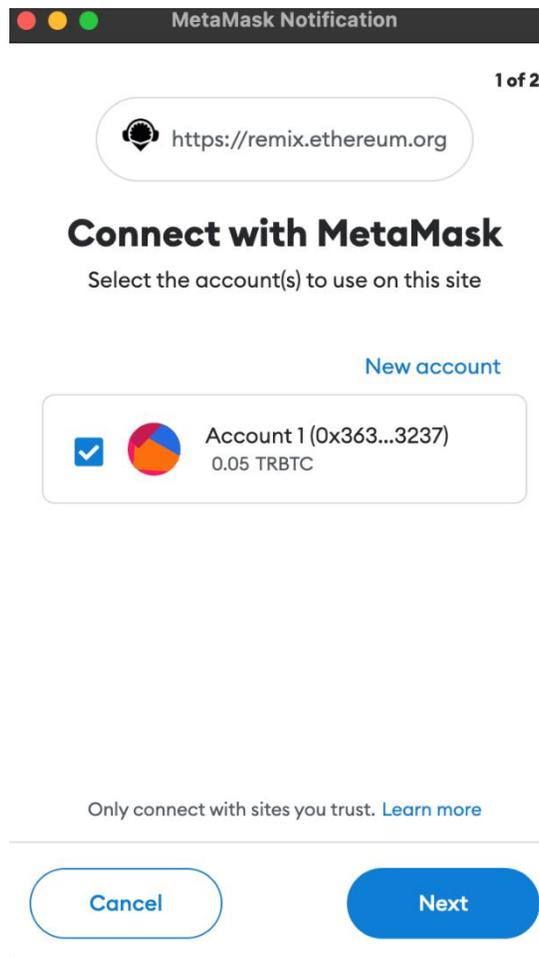
With the RSK network selected at Metamask...

At Remix, on the left side, locate the button **Deploy and run transactions**.

At Environment, choose Injected Provider - MetaMask



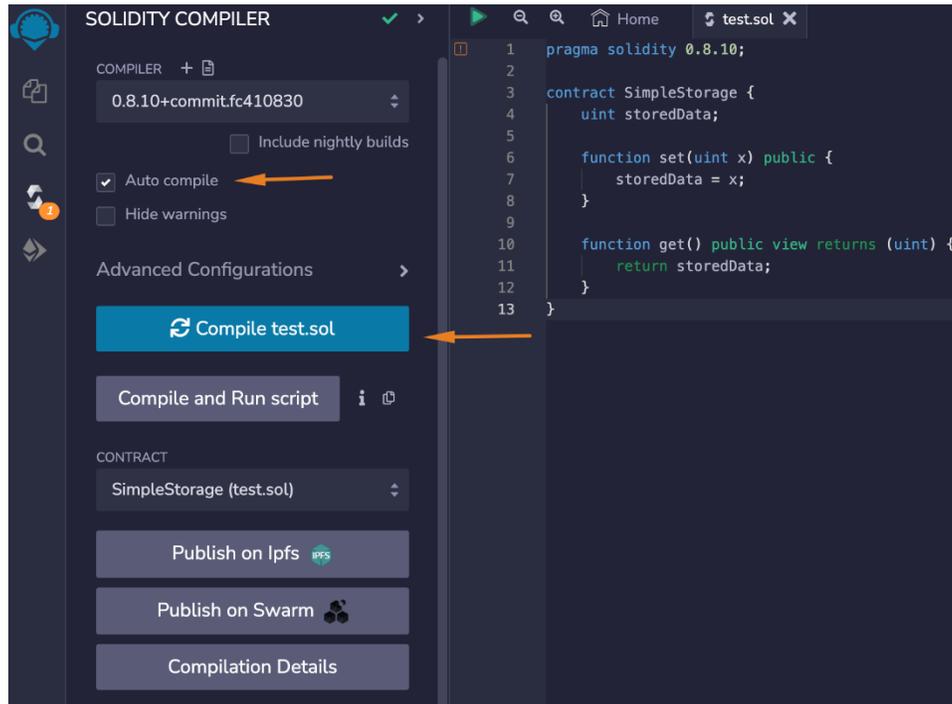
Injected Provider – MetaMask connects Remix with your active account in Metamask.



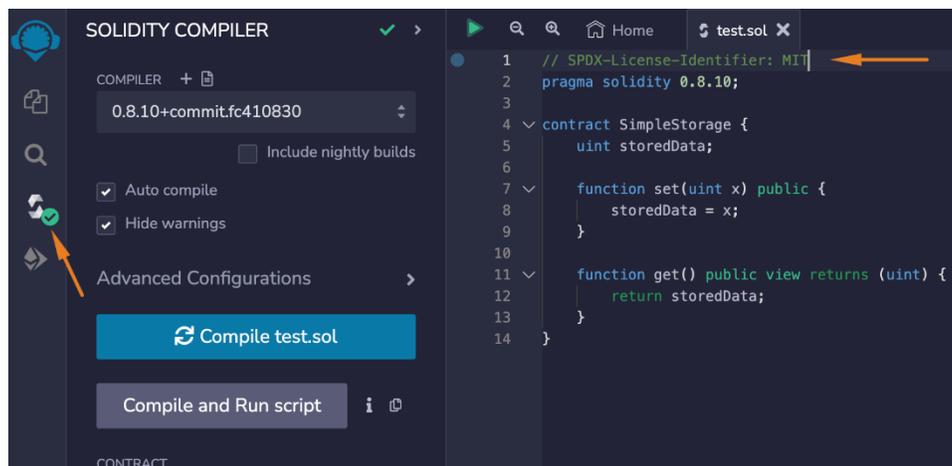
2.6 Compile your Smart Contract

Find the 3rd button at the left side and click on the Solidity compiler (make sure to enable auto-compile, it will save you a lot of time!)

Click the button Compile test.sol



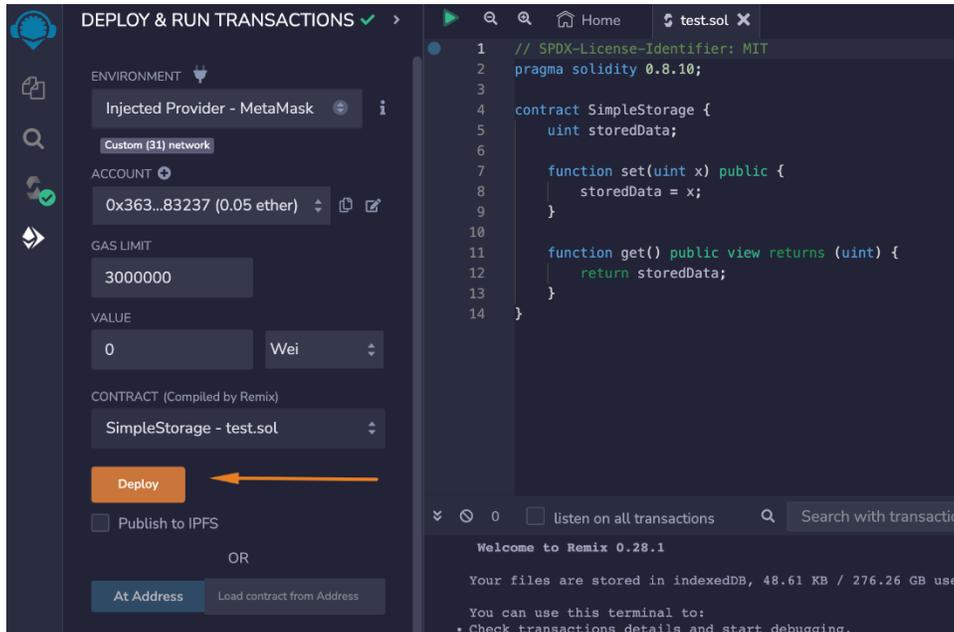
Check the green sign that will give you the message of successful compilation.



Tip: To prevent warnings add comment to line 1: `// SPDX-License-Identifier: MIT`

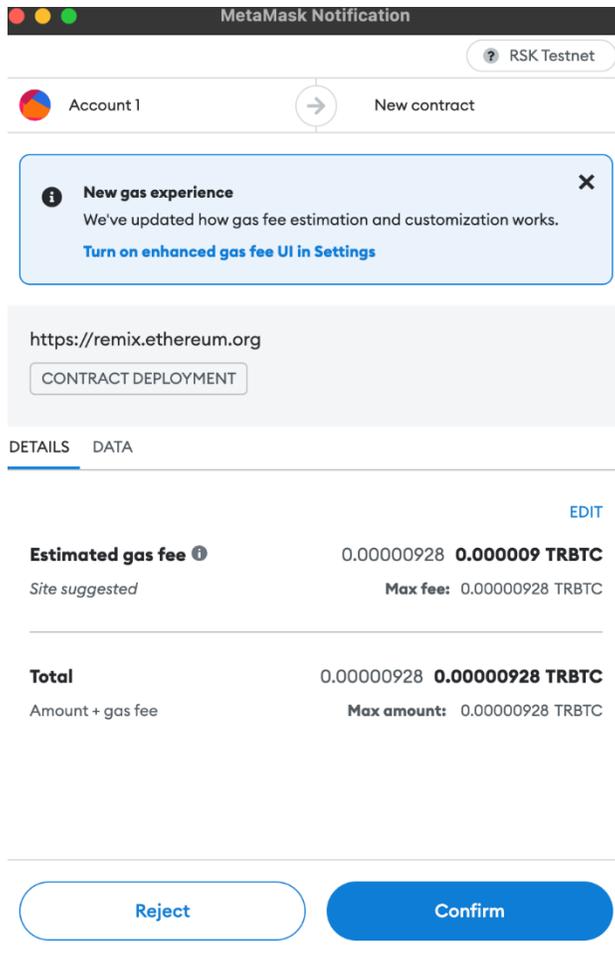
2.7 Deploy a Smart Contract on the RSK TESTNET

On the left side panel, go to the button **Deploy and run transactions**.



For now we have only one smart contract, so it is automatically selected in the dropdown.

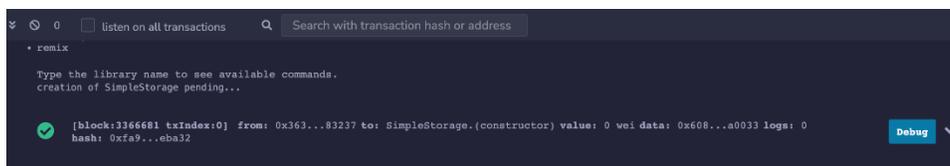
It will open a Metamask popup window, to confirm the transaction and create the smart contract test.sol



Click Confirm.

At bottom right, we will get the message **creation of SimpleStorage pending...**

Once it is confirmed, we can have a closer look at it.



Click on the transaction line or the debug button (at the right side) to see more details of the transaction.

```
[block:336681 txIndex:0] from: 0x363...83237 to: SimpleStorage.(constructor) value: 0 wei data: 0x608...a033 logs: 0
hash: 0xfa9...eba32 Debug ^

status
  true Transaction mined and execution succeed

transaction hash
  0x91b6a1e6fe33dfcf33b930d62b41c3e62cde777ae3370a37eb1936008d93fc51

from
  0x363A47420FDAd118a15Af0a58d87149997D83237

to
  SimpleStorage.(constructor)

gas
  142397 gas

transaction cost
  142397 gas

input
  0x608...a033

decoded input
  ()
```

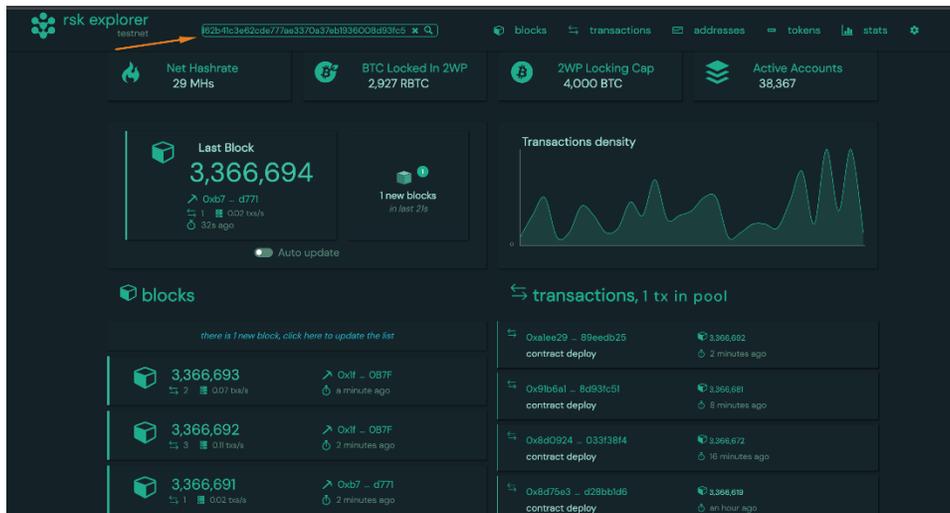
Copy the transaction hash to verify using the RSK blockchain explorer

In this example, the transaction hash is:

0x91b6a1e6fe33dfcf33b930d62b41c3e62cde777ae3370a37eb1936008d93fc51

2.8 RSK Explorer

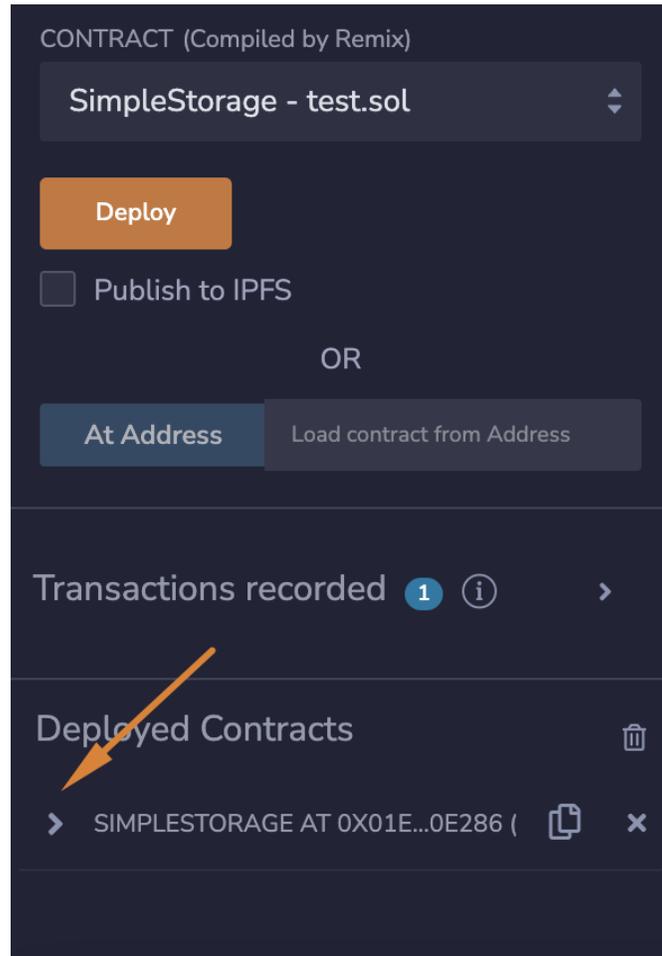
The RSK explorer is the blockchain explorer for RSK transactions. We will use the [Testnet explorer](#)

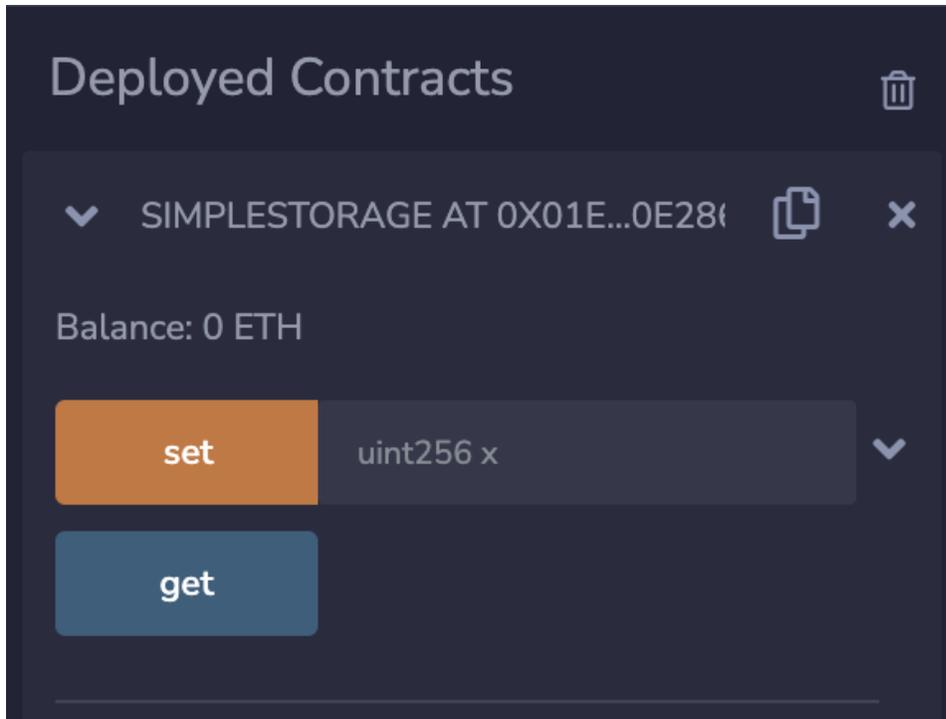


This is what we would expect to see:

2.9 Interact with your Smart Contract

When a smart contract is deployed with Remix, we can see it in the left panel under deploy and run transactions:





These are the same functions we created in our smart contract!

The orange buttons are functions which will change some information stored on the blockchain.

Every time we call it, the function will expend some gas and will perform what it was programmed to do.

The blue buttons are functions which are read-only and do not change anything stored on the blockchain, they only fetch data. We do not expend gas when using them, only the function that change things on the blockchain have a cost for us.

Get the value from the Blockchain

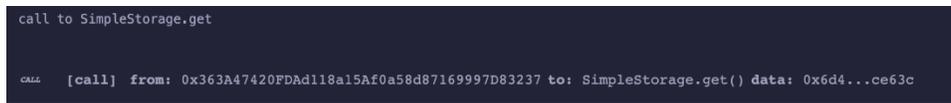
First of all, we will check the value stored at the time of deployment.

Click the button get



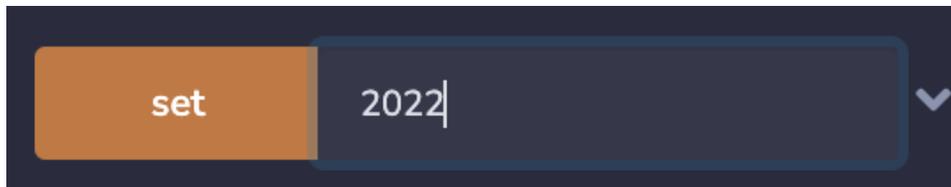
We do not have any value stored, because we have not yet defined anything.

At the bottom right, we can see that a call to the `SimpleStorage.get()` function was made.

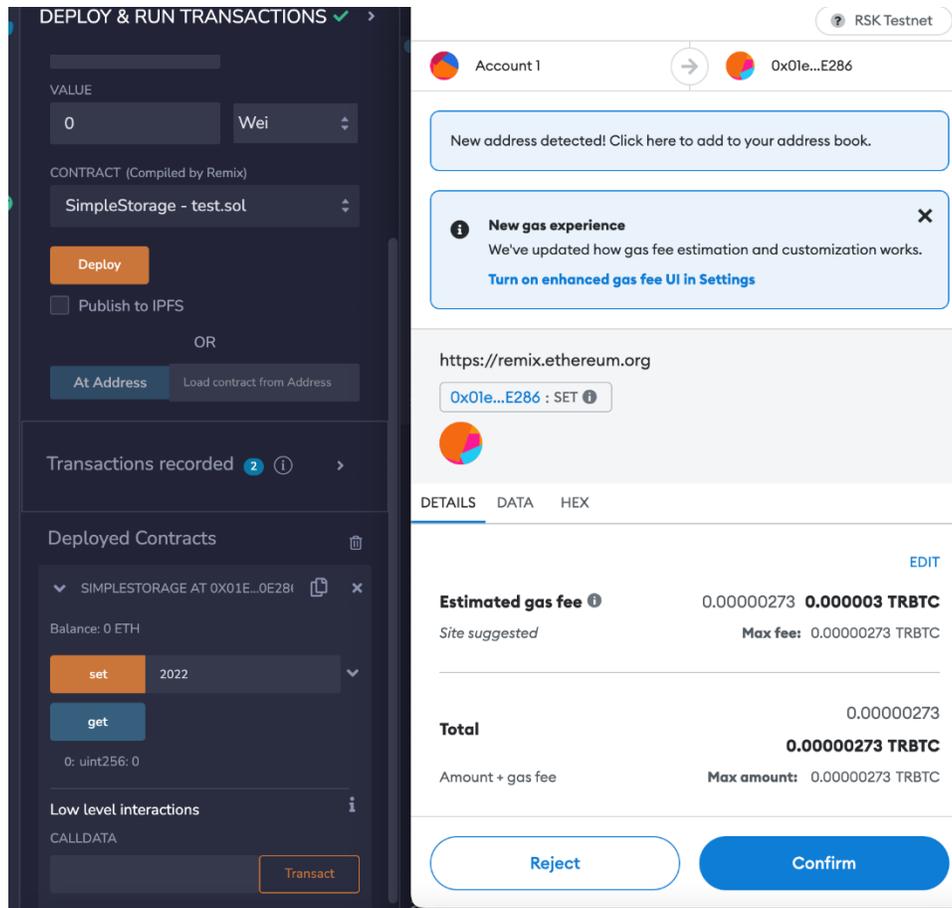


Set the value on the Blockchain

Put a value in the field at the right side of the set button, and click on the button



A Metamask popup window will open, to confirm the transaction to store the value on the Blockchain. The confirmation is needed because as described earlier, now we will make a change on the blockchain and we will have to spend some gas to do it.

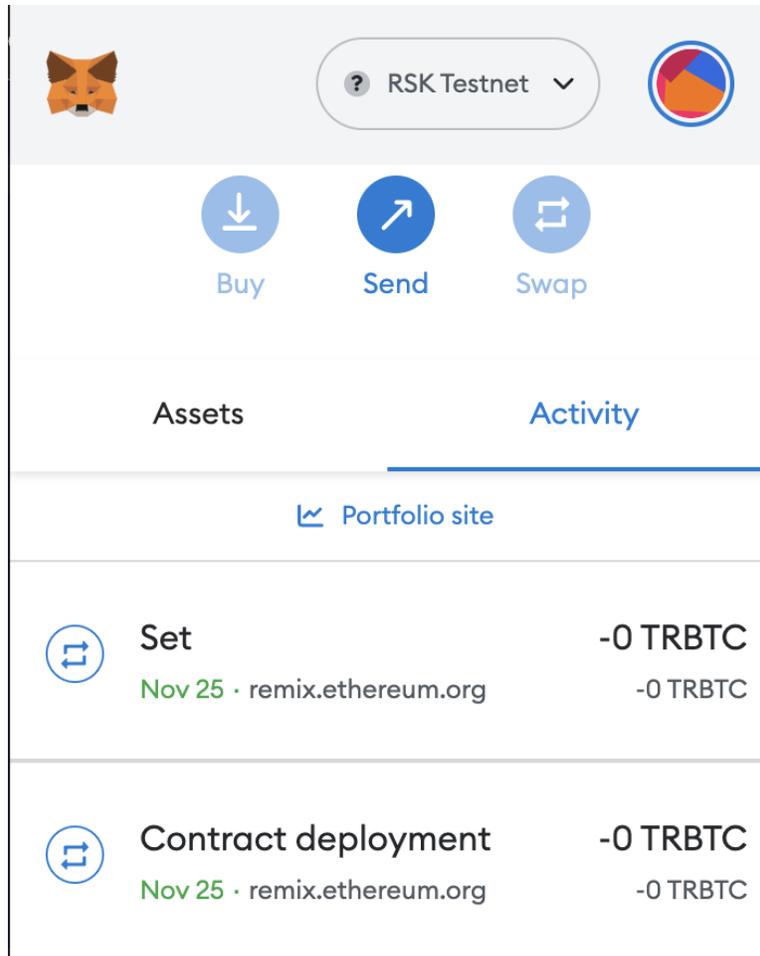


Click on Confirm

At the bottom right, we can verify that the transaction is pending, awaiting confirmation at blockchain:

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

After a few seconds, Metamask will show when the transaction has been confirmed!



The screenshot shows a mobile wallet interface for RSK Testnet. At the top, there is a fox icon, a dropdown menu set to 'RSK Testnet', and a circular icon with red, blue, and yellow segments. Below this are three circular buttons: 'Buy' (down arrow), 'Send' (up arrow), and 'Swap' (two arrows). A horizontal line separates these from two tabs: 'Assets' and 'Activity', with 'Activity' being the active tab. Below the tabs is a link labeled 'Portfolio site'. The main content area displays a list of transactions:

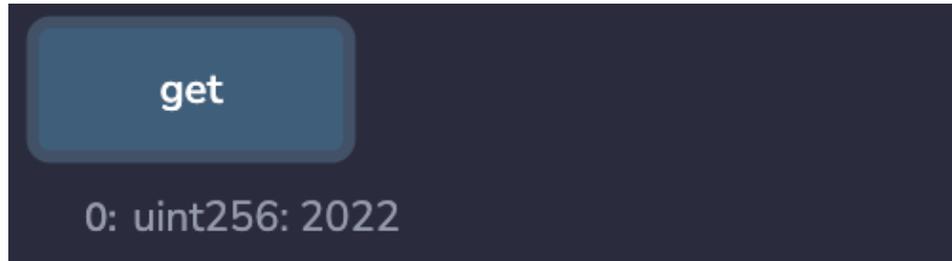
Transaction Type	Date & Source	Amount
Set	Nov 25 · remix.ethereum.org	-0 TRBTC
Contract deployment	Nov 25 · remix.ethereum.org	-0 TRBTC

Click on Set to see the transaction details

Get (again)

Now that we have the value 2022 saved, and we can re-query from the blockchain.

Click the button get once more.



The returned value is correct!

Where do I go from here?

It all depends on your business and your imagination. Smart contracts can be used in registering and securing financial transactions, they can be used to represent sellable items in games, secure legal documents and decisions or serve as digital deeds in the real estate space. The sky is the limit.

3. Knowledge Assessment

What are the fees in blockchain (tick all that apply)?

- Small amounts of cryptocurrency required to process a transaction
- No fees are required to use blockchain
- Fees are paid to the network for storage
- Large amounts of cryptocurrency required to execute a contract

4. References

- Solidity Documentation: The official documentation for Solidity, the most popular programming language for building smart contracts on the Ethereum blockchain.
- Ethereum.org: The official website of the Ethereum blockchain has a great developer section with tutorials, documentation, and other resources.
- Mastering Ethereum: Building Smart Contracts and DApps by Andreas M. Antonopoulos and Gavin Wood: A comprehensive guide that covers the basics of the Ethereum blockchain, smart contracts, and how to build decentralized applications.
- Programming the Blockchain in C#: A Comprehensive Guide to Building Decentralized Applications by Nicolas Dorier: This book provides an introduction to blockchain technology, a practical guide to building decentralized applications, and a comprehensive tutorial on the C# programming language.
- Building Ethereum DApps: Decentralized Applications on the Ethereum Blockchain by Roberto Infante: This book is a practical guide to building decentralized applications on the Ethereum blockchain.
- Building Blockchain Projects: Building Decentralized Blockchain Applications with Ethereum and Solidity by Narayan Prusty: This book is a step-by-step guide to building real-world decentralized applications on the Ethereum blockchain.
- Ethereum Smart Contract Development: Build blockchain-based decentralized applications using Solidity by Mayukh Mukhopadhyay: This book covers the basics of Solidity programming and provides practical examples of building smart contracts on the Ethereum blockchain.