



@maryemhadjwannes

Create **ERC-20** Token in 10 mins

Start Tutorial





REMIX

<https://remix.ethereum.org/>



METAMASK

<https://metamask.io/download/>

No local installation is needed; everything is online!

How Tokens Work in Ethereum Virtual Machine

Tokens on Ethereum act as a **ledger** or **database** that keeps track of addresses and their corresponding balances. This database ensures secure and transparent transfer of tokens between addresses.

- **Visualizing the Token Ledger**

The token ledger can be imagined as a table with two columns:

Address	Balance
0x123	100
0x456	0
0x789	0

- **Example Transactions**

1. Transfer 25 Tokens from 0x123 to 0x456

The ledger updates as follows:

Address	Balance
0x123	75
0x456	25
0x789	0

2. Transfer 70 Tokens from 0x123 to 0x789

The ledger updates again:

Address	Balance
0x123	5
0x456	25
0x789	70

Key Takeaways

- **Each address represents a user or wallet.**
- **The balance column reflects the tokens each address holds.**
- **Transactions update balances by deducting tokens from the sender and adding them to the recipient.**
- **This simple structure is maintained and enforced by the Ethereum network, ensuring secure and tamper-proof token management.**

OpenZeppelin

OpenZeppelin is a renowned company specializing in security audits for blockchain projects. They also provide a suite of **free, secure, and audited smart contracts** that anyone can use with confidence.



Their contracts are:

- **Pre-audited:** No need to worry about vulnerabilities.
- **Standard-Compliant:** Built to adhere to widely accepted token standards (like ERC20, ERC721, etc.).

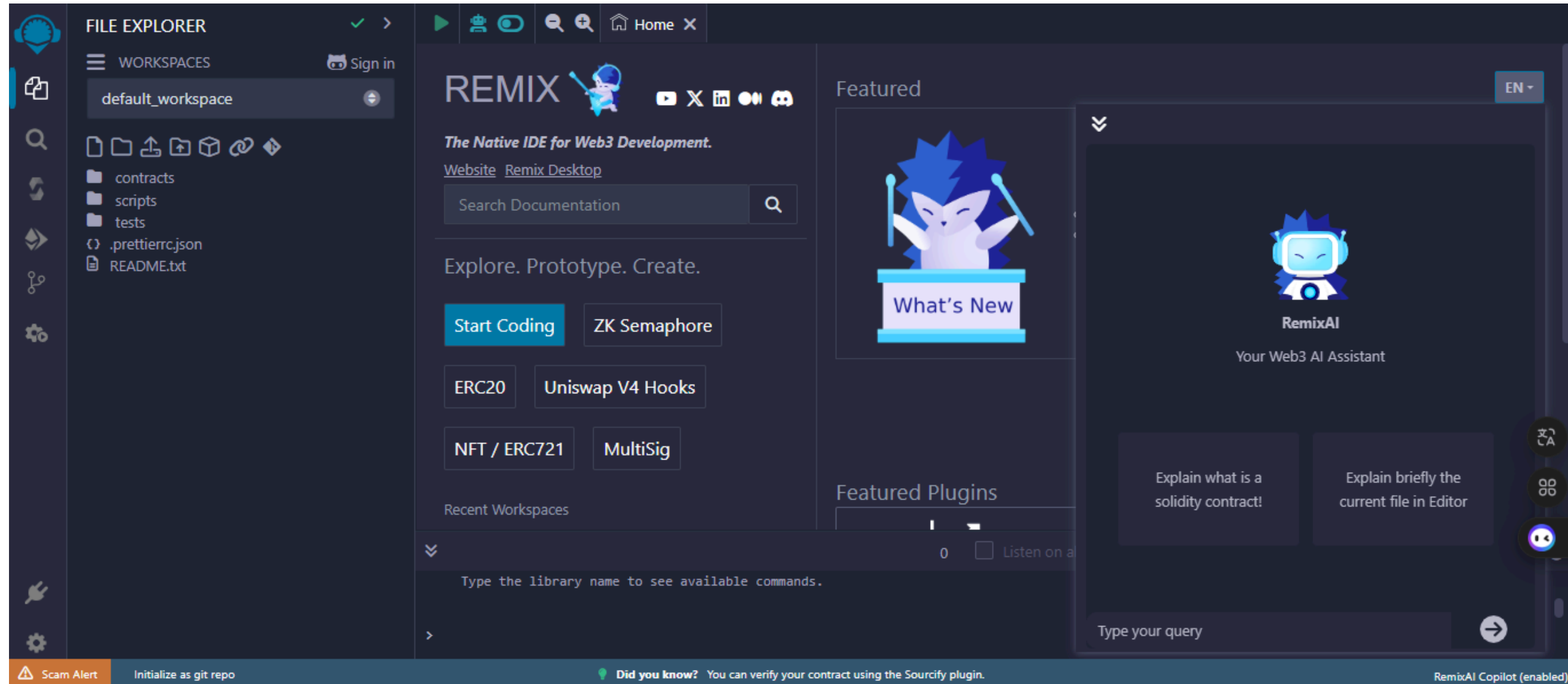
By using OpenZeppelin's contracts, you save time and ensure your token meets industry best practices. **For this tutorial, we'll focus on the ERC20 contract.**

Explore their free contracts here: [OpenZeppelin GitHub](#)

●●● Create Token

@maryemhadjwannes

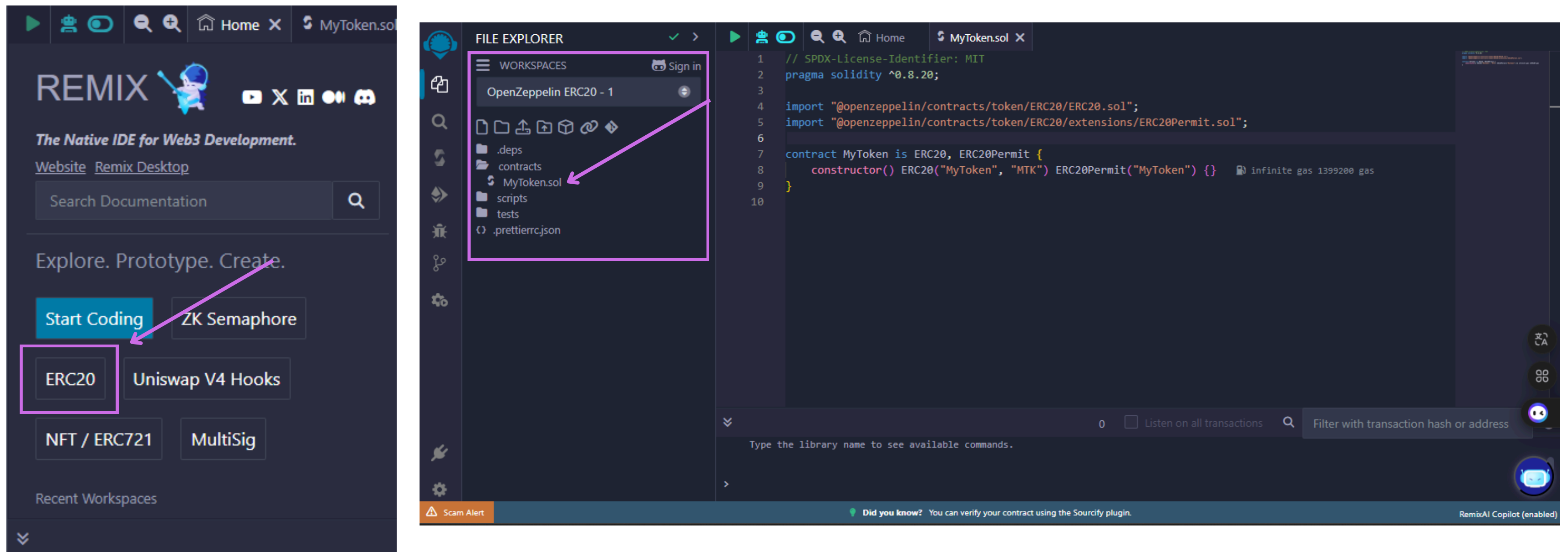
1- Open Remix IDE : This is the default page



●●● Create Token

@maryemhadjwannes

2- Choose **ERC20** directly, and it will take you to the **OpenZeppelin ERC20 workspace** and create a **MyToken.sol** file with the necessary imports. You can create the file manually, but this method automates the imports and initial setup.



Code Breakdown

1- License Declaration

```
// SPDX-License-Identifier: MIT
```

- Specifies the license type for the contract, which in this case is MIT, a permissive open-source license.

2- Solidity Version

```
pragma solidity ^0.8.20;
```

- Indicates the version of Solidity required to compile the contract, ensuring compatibility and access to specific language features.

Code Breakdown

3- Import Statements

```
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";  
import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Permit.sol";
```

- **ERC20.sol:** Contains the base implementation of the ERC20 token standard, providing functionality like transfers, balance tracking, and allowance mechanisms.
- **ERC20Permit.sol:** Adds support for "permit" functionality, allowing approvals via digital signatures (EIP-2612) instead of requiring on-chain transactions.

4- Contract Declaration

```
contract MyToken is ERC20, ERC20Permit {
```

- Defines a new contract called **MyToken** that inherits functionality from both **ERC20** and **ERC20Permit**.

Code Breakdown

5- Constructor

```
constructor() ERC20("MyToken", "MTK") ERC20Permit("MyToken") {}
```

- The constructor initializes the **ERC20** and **ERC20Permit** parent contracts:
 - **ERC20:**
 - **"MyToken"**: The name of the token.
 - **"MTK"**: The symbol of the token.
 - **ERC20Permit:**
 - **"MyToken"**: The name of the token used in the permit mechanism.

●●● Create Token

@maryemhadjwannes

3- In this step, we extend our token contract with a custom minting function `minHundred()`. This function allows the caller to mint 100 tokens directly into their wallet

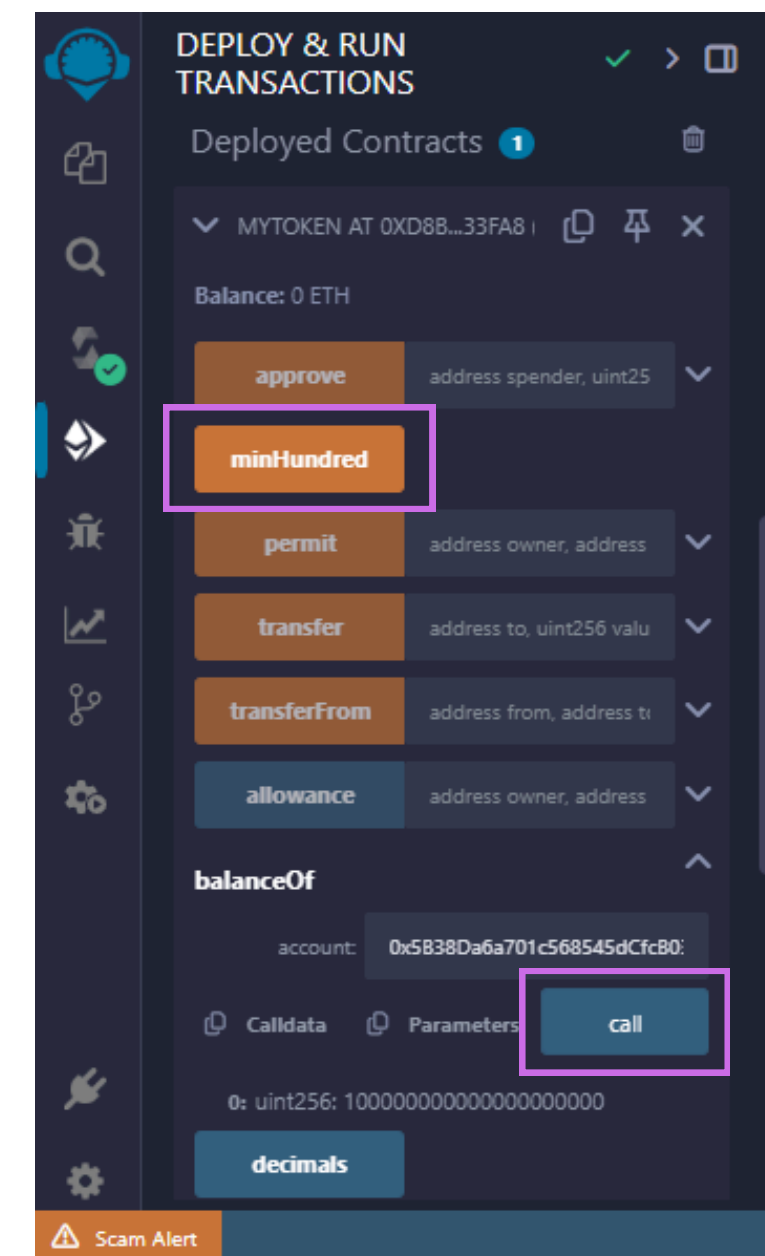
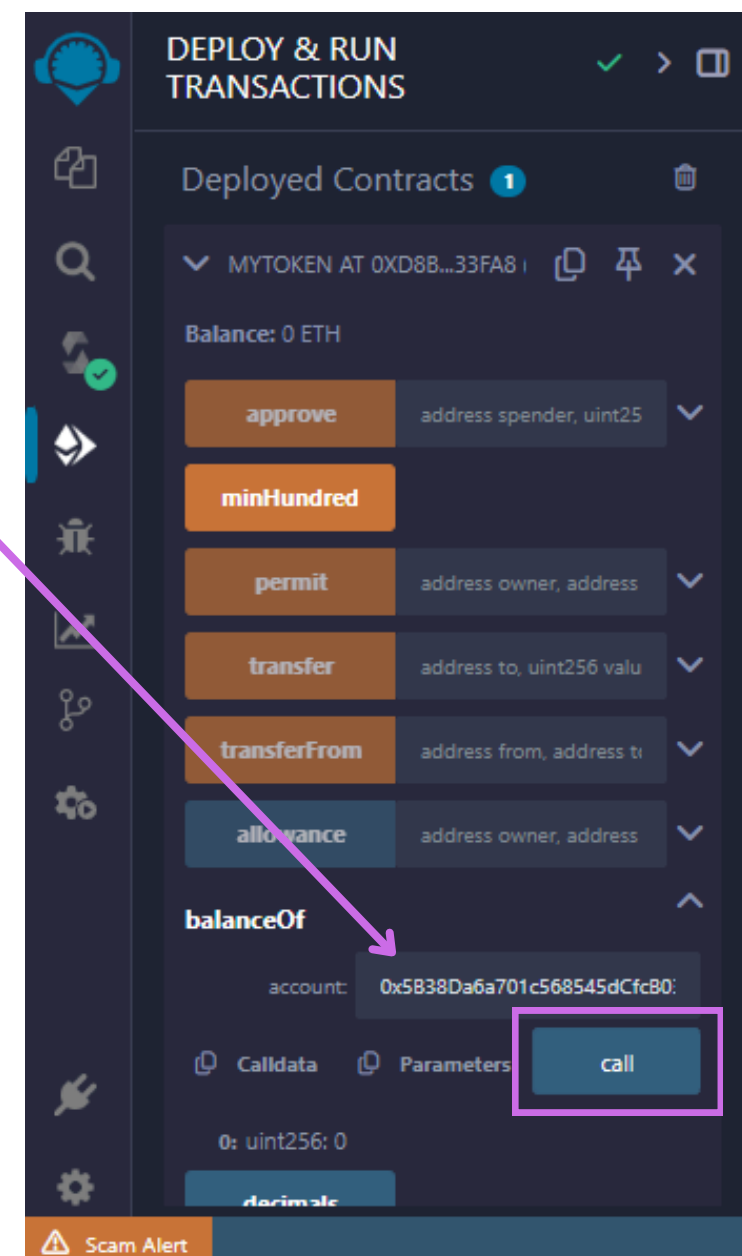
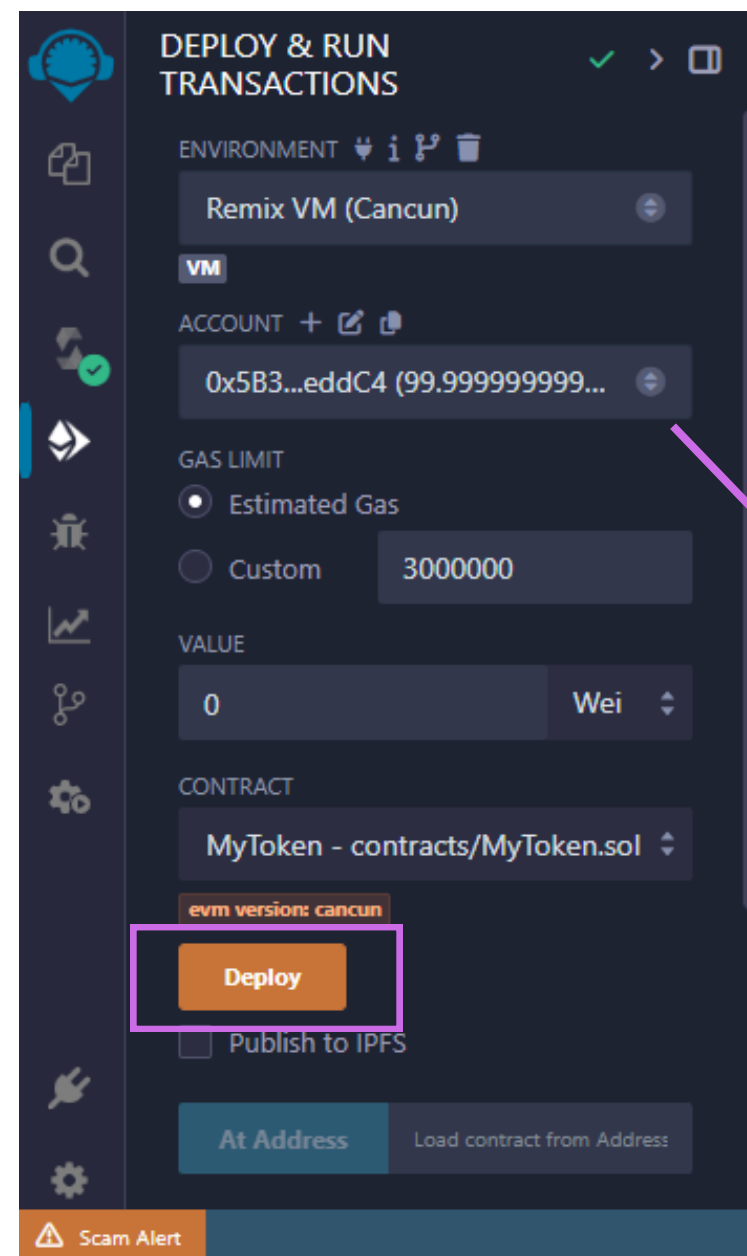
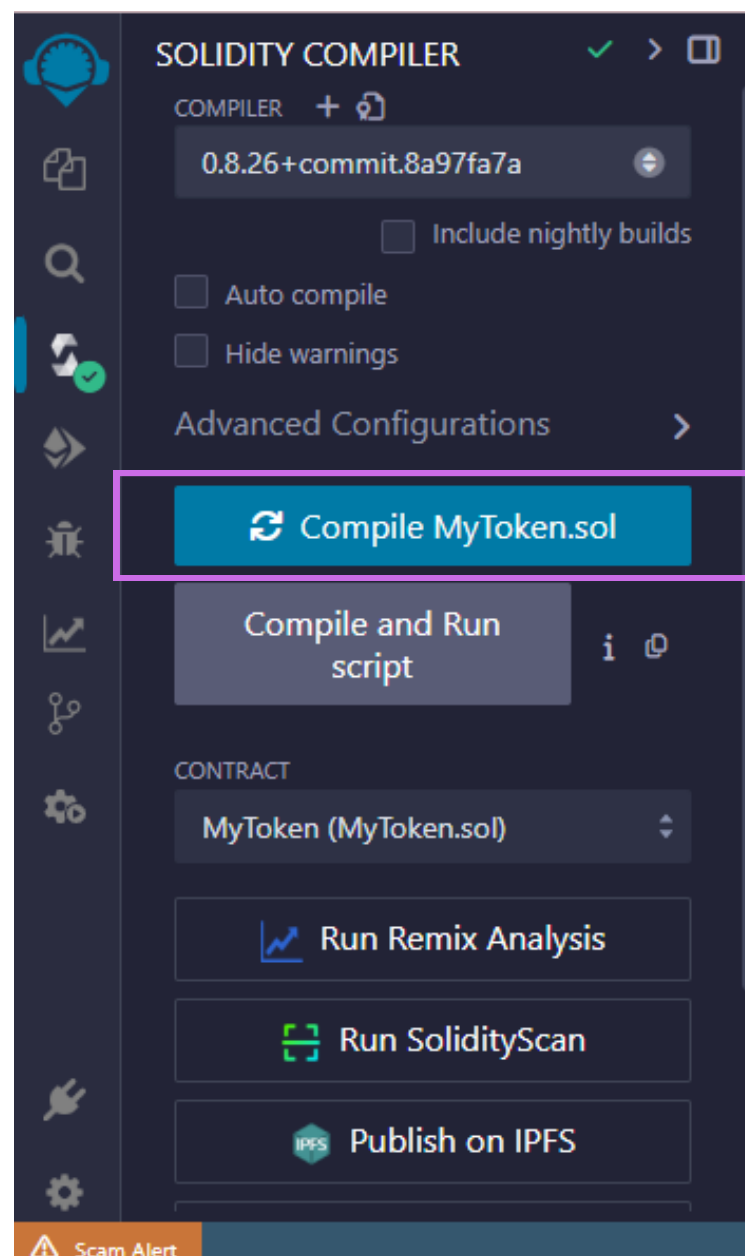
```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.20;
3
4 import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
5 import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Permit.sol";
6
7 contract MyToken is ERC20, ERC20Permit {
8     constructor() ERC20("MyToken", "MTK") ERC20Permit("MyToken") {}
9
10    function minHundred() public {
11        _mint(msg.sender, 100 * 10**18);
12    }
13 }
14
```

- **Public Access:** Any user can call this function.
- **Minting:** Creates 100 tokens, considering the default 18 decimals.
- **Assignment:** Tokens are sent to the caller's address (**msg.sender**).

●●● Create Token

@maryemhadjwannes

4- **Compile** and **deploy** the contract. In the deployed contracts section, you will find functions that you can test, such as **balanceOf**. Enter your address (from the **Accounts** section) into the **balanceOf** function, and it will initially return 0. Then, call the **mintHundred** function and check **balanceOf** again to see the new balance updated to **100 tokens**.



●●● Create Token

@maryemhadjwannes

5- Currently, anyone can call the `mintHundred` function to mint 100 tokens for themselves, which is **not ideal for controlling token supply**. To fix this, **we can restrict the function so that only the contract owner can mint tokens**.

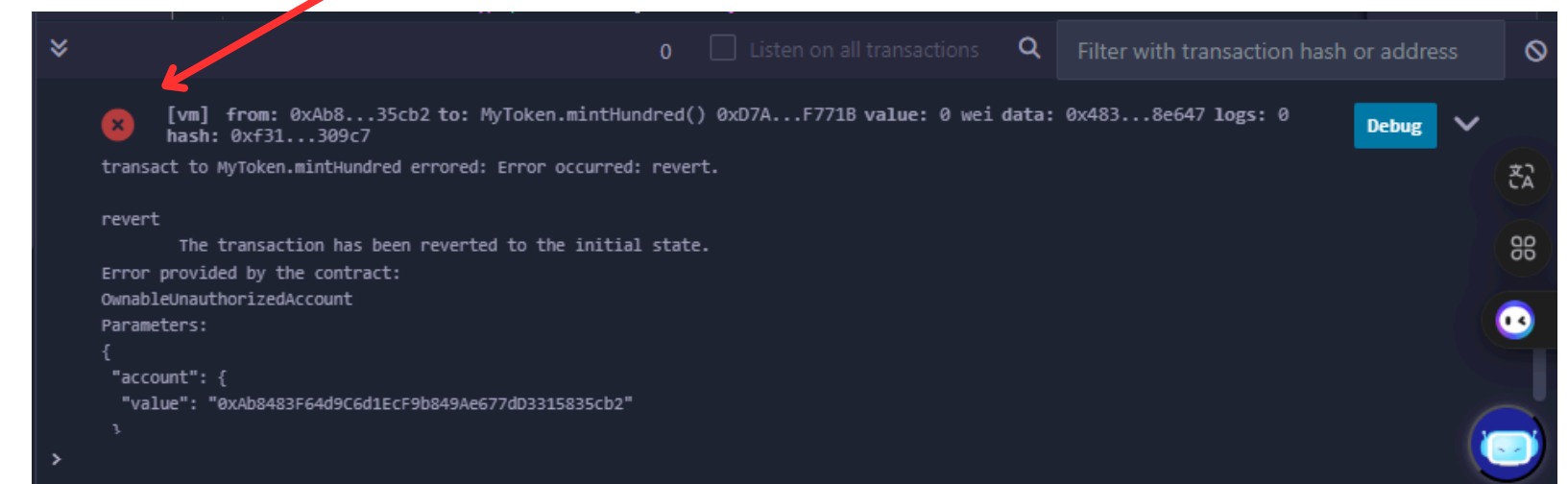
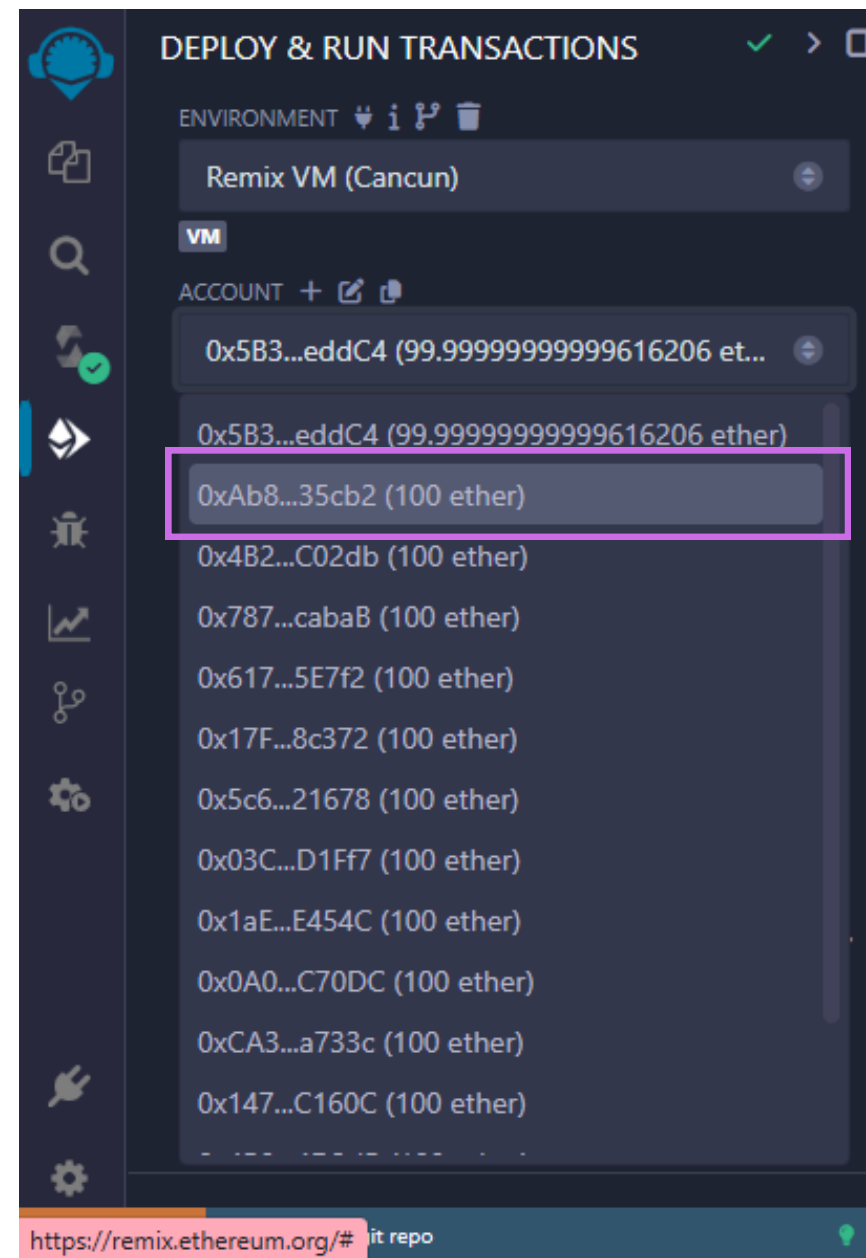
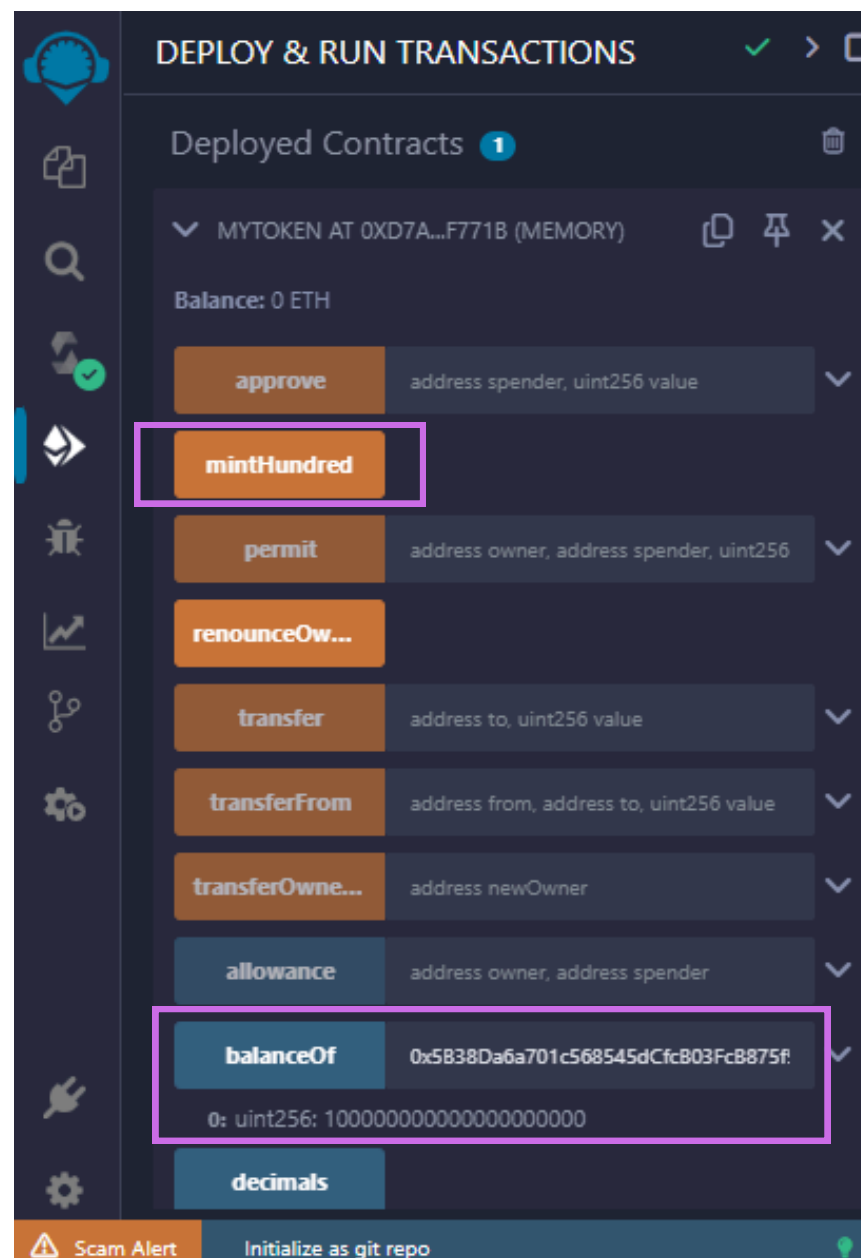
ERC20 contracts include a modifier called `onlyOwner`. This modifier **allows us to restrict access to specific functions**, ensuring **only the owner can execute them**. By adding this, we secure the `mintHundred` function and prevent unauthorized minting.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.20;
3
4 import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
5 import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Permit.sol";
6 import "@openzeppelin/contracts/access/Ownable.sol";
7
8 contract MyToken is ERC20, ERC20Permit, Ownable {
9
10     constructor(address initialOwner) ERC20("MyToken", "MTK") ERC20Permit("MyToken")
11         Ownable(initialOwner) {}
12
13     function mintHundred() public onlyOwner {    ⚙ infinite gas
14         _mint(msg.sender, 100 * 10**18);
15     }
16 }
```

●●● Create Token

@maryemhadjwannes

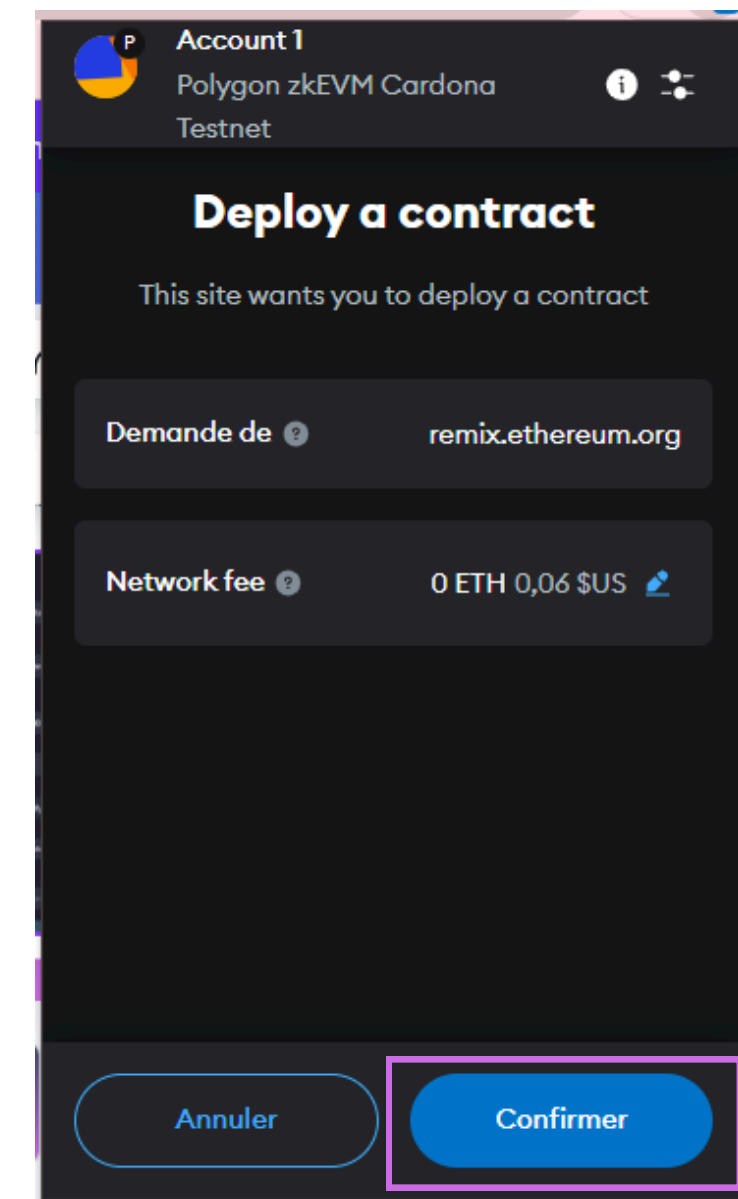
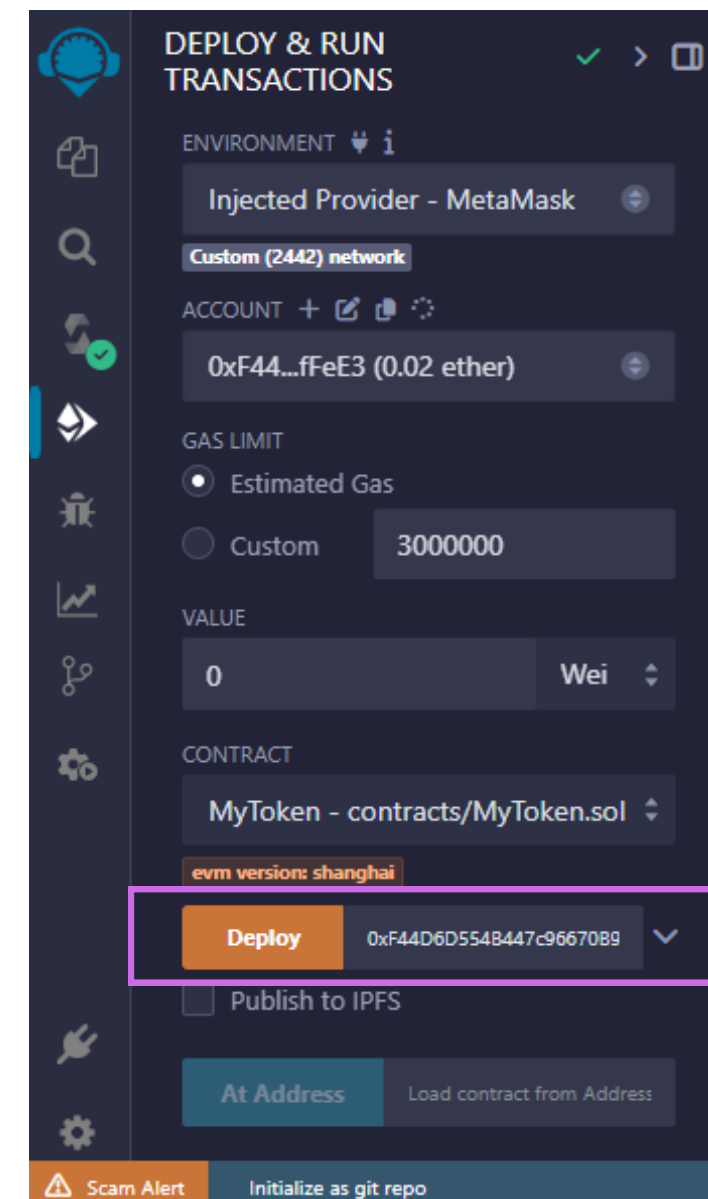
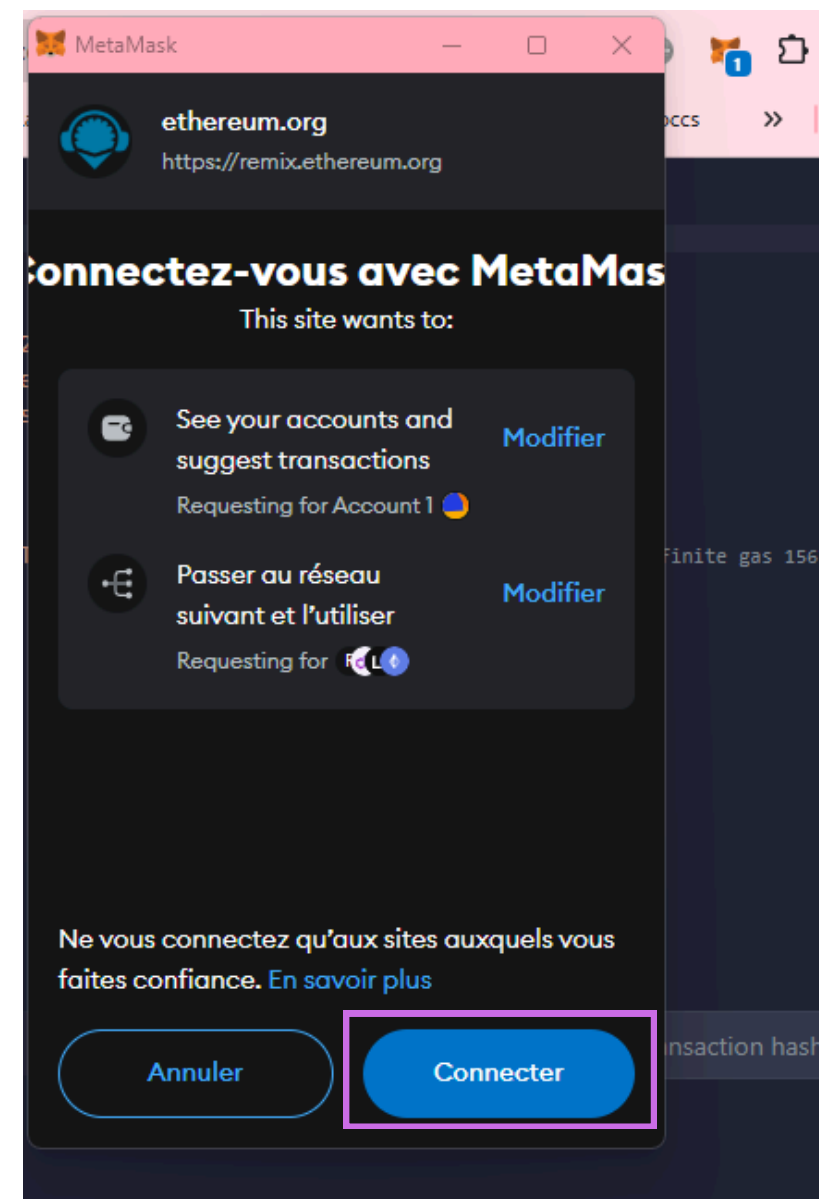
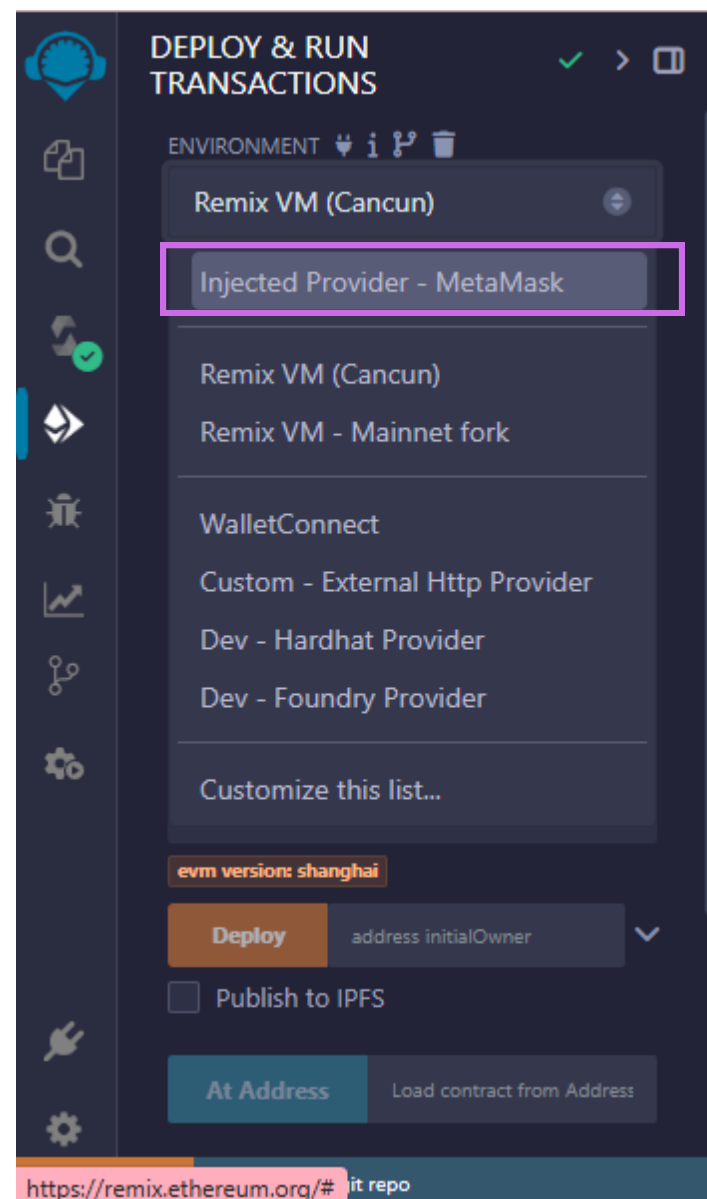
6- When we mint tokens for the owner of the contract, 100 tokens are successfully minted. However, if we try to switch to **another address** and attempt to mint, it will return an **error** due to the **onlyOwner** modifier, which restricts the minting function to the contract owner only.



●●● Create Token

@maryemhadjwannes

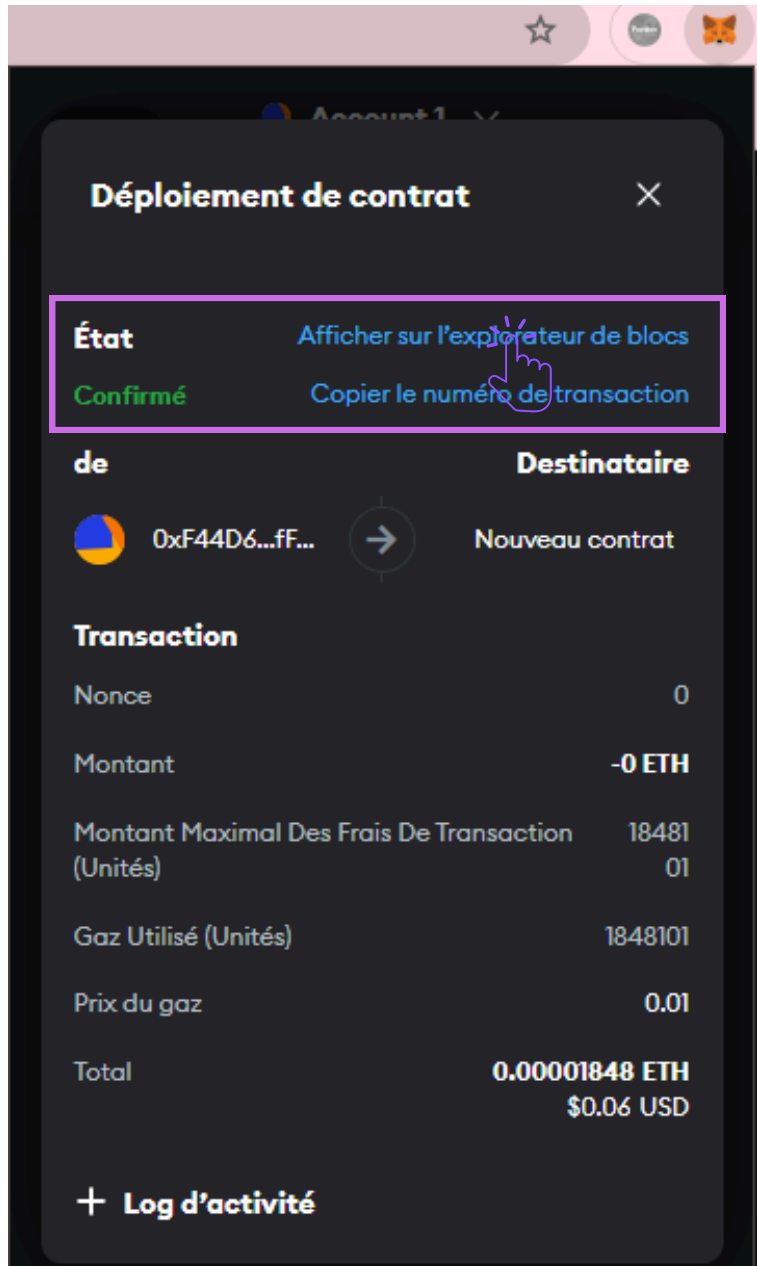
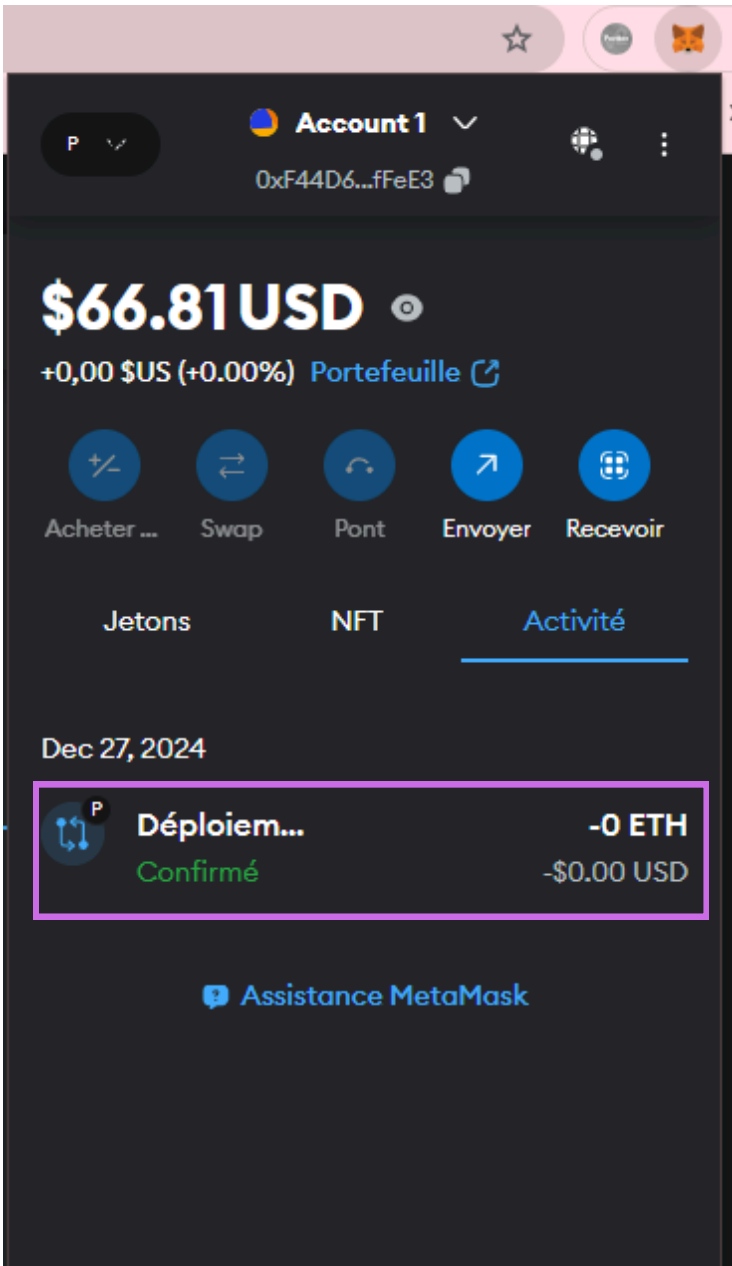
7- Now, let's update the web page. Go to the **Deploy** section, select Injected Provider (MetaMask). MetaMask will pop up, prompting you to connect your wallet. After connecting, click Deploy. MetaMask will ask for confirmation to deploy the contract, and once approved, your token will be successfully launched!



●●● Create Token

@maryemhadjwannes

8- In MetaMask's transaction history, you can click on the deployment transaction. This will take you to [PolygonScan \(on the testnet\)](#), where you can view detailed information about the transaction, including the [deployment transaction fee](#) and the [address that created the contract](#).



Cardona Testnet

Search by Address / Txn Hash / Bloc

OverviewLogs (1)State

[This is a Polygon zkEVM Cardona Testnet transaction only]

Transaction Hash:	0xda4753b98cf009b3a7d3245182b5cc2b0d13d5ed5d5418f0c48d8a95c90bd4dd
Status:	Success
Block:	9410758 Confirmed by Sequencer
Timestamp:	8 mins ago (Dec-27-2024 01:13:57 AM UTC)
Transaction Action:	Call 0x61016060 Method by 0xF44D6D55...4ABeFeE3
From:	0xF44D6D554B447c96670B9F81b4BACe94ABeFeE3
To:	[0xc3cc39a225aa2a2488acc1c1a551f52062d8a7a4 Created]
Value:	0 ETH
Transaction Fee:	0.00001848101 ETH
Effective Gas Price:	0.01 Gwei (0.000000000001 ETH)

●●● Create Token

@maryemhadjwannes

9- Since PolygonScan doesn't recognize it as a token yet, return to Remix and call the **mintHundred** function to **mint 100 tokens**.

The screenshot displays the Remix IDE interface for deploying and interacting with a token contract. The left sidebar shows the 'DEPLOY & RUN TRANSACTIONS' panel with the 'mintHundred' function highlighted. The central editor shows the Solidity code for the 'MyToken' contract, which includes a constructor and a 'mintHundred' function. A MetaMask transaction confirmation window is overlaid, showing the transaction details and a 'Confirmer' button. The right sidebar shows the account balance and transaction history, with the 'Interaction...' transaction highlighted.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.20;
3
4 import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
5 import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Permit.sol";
6 import "@openzeppelin/contracts/access/Ownable.sol";
7
8 contract MyToken is ERC20, ERC20Permit, Ownable {
9
10     constructor(address initialOwner) ERC20("MyToken", "MyToken") Ownable(initialOwner) {}
11
12     function mintHundred() public onlyOwner {
13         _mint(msg.sender, 100 * 10**18);
14     }
15 }
16
```

Transaction Confirmation Details:

- Demande de : remix.ethereum.org
- Interagir avec : 0xc3Cc3...8A7a4
- Network fee : 0 ETH 0,00 \$US

Transaction History:

Transaction	Amount	Status
Interaction...	-0 ETH	Confirmé
Déploiement...	-0 ETH	Confirmé

Transaction Confirmation Message:

Transaction confirmée
La transaction 1 a été confirmée ! Consulter sur Cardona Zkevm Polygonscan

10- **After minting 100 tokens** using the mintHundred function, PolygonScan will show that 100 tokens were added to your address. This usually appears as if the tokens are coming from the null address (0x0...0), and it will display the token name as **MyToken**, the symbol as **MTK**, and the type as **ERC20**.

Cardona Testnet

Search by Address / Txn Hash / Block / Token

[This is a Polygon zkEVM Cardona Testnet transaction only]

Transaction Hash:

0xb5eb720cf63f84f8cb1694bca421e8568e3add4a35a8e18f73e2fa1fe7cffa53

Status:

Success

Block:

9411205 Confirmed by Sequencer

Timestamp:

2 mins ago (Dec-27-2024 01:37:53 AM UTC)

From:

0xF44D6D554B447c96670B9F81b4BACe94ABeFeE3

Interacted With (To):

0xc3Cc39a225aa2A2488Acc1C1a551f52062D8A7a4

ERC-20 Tokens Transferred:

All Transfers Net Transfers

From 0x00000000...00000000 To 0xF44D6D55...4ABeFeE3 For 100 ERC-20: MyToken (MTK)

Value:

0 ETH

Transaction Fee:

0.00000070207 ETH

Effective Gas Price:

0.01 Gwei (0.000000000001 ETH)

●●● Create Token

@maryemhadjwannes

11- To display more information, **the total supply of our token is 100 MTK**, with **1 holder (my address)** and **1 total transfer (the minting transaction)**. This shows that the minting process has successfully created 100 tokens and assigned them to the first holder. And of course, the market value is 0 ETH for now, Haha! But hey, it's a start! 😊

Cardona Testnet

Search by Address / Txn Hash / Block / Token

ERC-20

Overview

MAX TOTAL SUPPLY
100 MTK

HOLDERS
1

TOTAL TRANSFERS
1

Market

PRICE
\$0.00 @ 0.000000 ETH

ONCHAIN MARKET CAP ⓘ
\$0.00

CIRCULATING SUPPLY MARKET CAP
-

Other Info

TOKEN CONTRACT (WITH 18 DECIMALS)
0xc3cc39a225aa2a2488acc1c1a551f52062d8a7a4

Transfers

Holders

Contract

⌵ A total of 1 transaction found

Download Page Data

First

<

Page 1 of 1

>

Last

Transaction Hash	Method ⓘ	Block	Age	From	To	Amount
0xb5eb720cf63...	0x4838e647	9411205	11 mins ago	0x00000000...000000000	0xF44D6D55...4ABeFeE3	100

First

<

Page 1 of 1

>

Last

●●● Create Token

@maryemhadjwannes

12- After transferring 25 MTK to another account, the token information updates accordingly. Now, there are 2 holders. The details show the amount held by each account: my first address holds 75 MTK, and the recipient holds 25 MTK.

Don't be surprised by 250000000000000000000000 because it's in Wei, which is equal to 25 MTK.

DEPLOY & RUN
TRANSACTIONS

✓

>

□

▼ MYTOKEN AT 0XC3C...8A7A4

📄

📌

✕

Balance: 0 ETH

⬆️⬆️

approve

address spender, uint256

▼

mintHundred

permit

address owner, address

▼

renounceOwn...

transfer

⬆️

to:

0x7c10730CC379188828377660

value:

2500000000000000000000

📄 Calldata

📄 Parameters

transact

transferFrom

address from, address to

▼

transferOwn...

address newOwner

▼

allowance

address owner, address

▼

Overview

MAX TOTAL SUPPLY
100 MTK

HOLDERS
2

TOTAL TRANSFERS
3

Market

PRICE
\$0.00 @ 0.000000 ETH

ONCHAIN MARKET CAP ⓘ
\$0.00

CIRCULATING SUPPLY MARKET CAP
-

Other Info

TOKEN CONTRACT (WITH 18 DECIMALS)
[0xc3cc39a225aa2a2488acc1c1a551f52062d8a7a4](#)

Transfers

Holders

Contract

A total of 3 transactions found

Download Page Data

First

<

Page 1 of 1

>

Last

Transaction Hash	Method	Block	Age	From	To	Amount
0xd80bdfd68a1...	Transfer	9411719	14 secs ago	0xF44D6D55...4ABeFeE3	→ 0x7c10730C...70680D098	25
0x79a96b1ffa9...	Transfer	9411531	10 mins ago	0xF44D6D55...4ABeFeE3	→ 0x7c10730C...70680D098	0.000000000000000025
0xb5eb720cf63...	0x4838e647	9411205	27 mins ago	0x00000000...00000000	→ 0xF44D6D55...4ABeFeE3	100



@maryemhadjwannes

I hope you found this tutorial helpful!