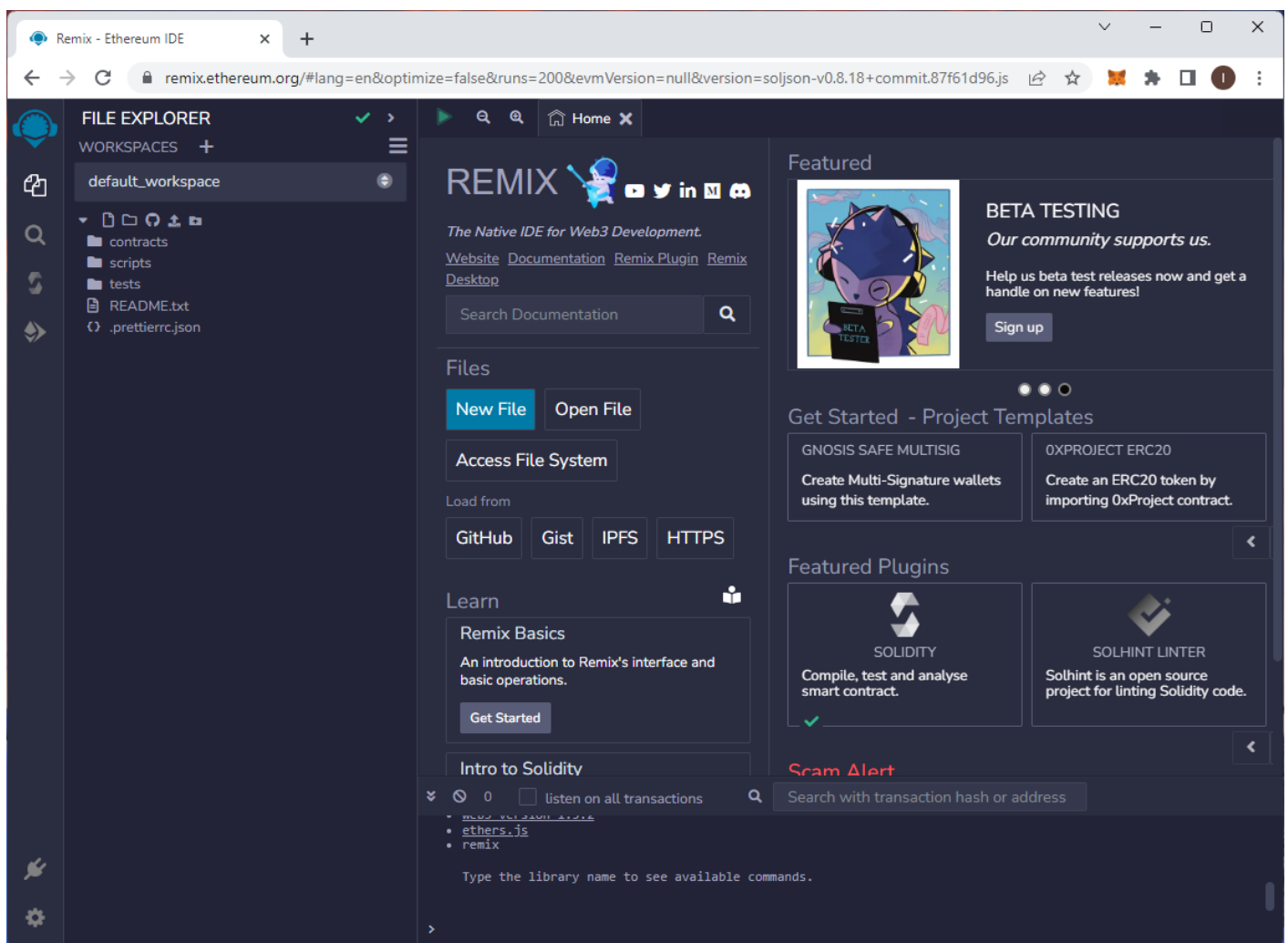
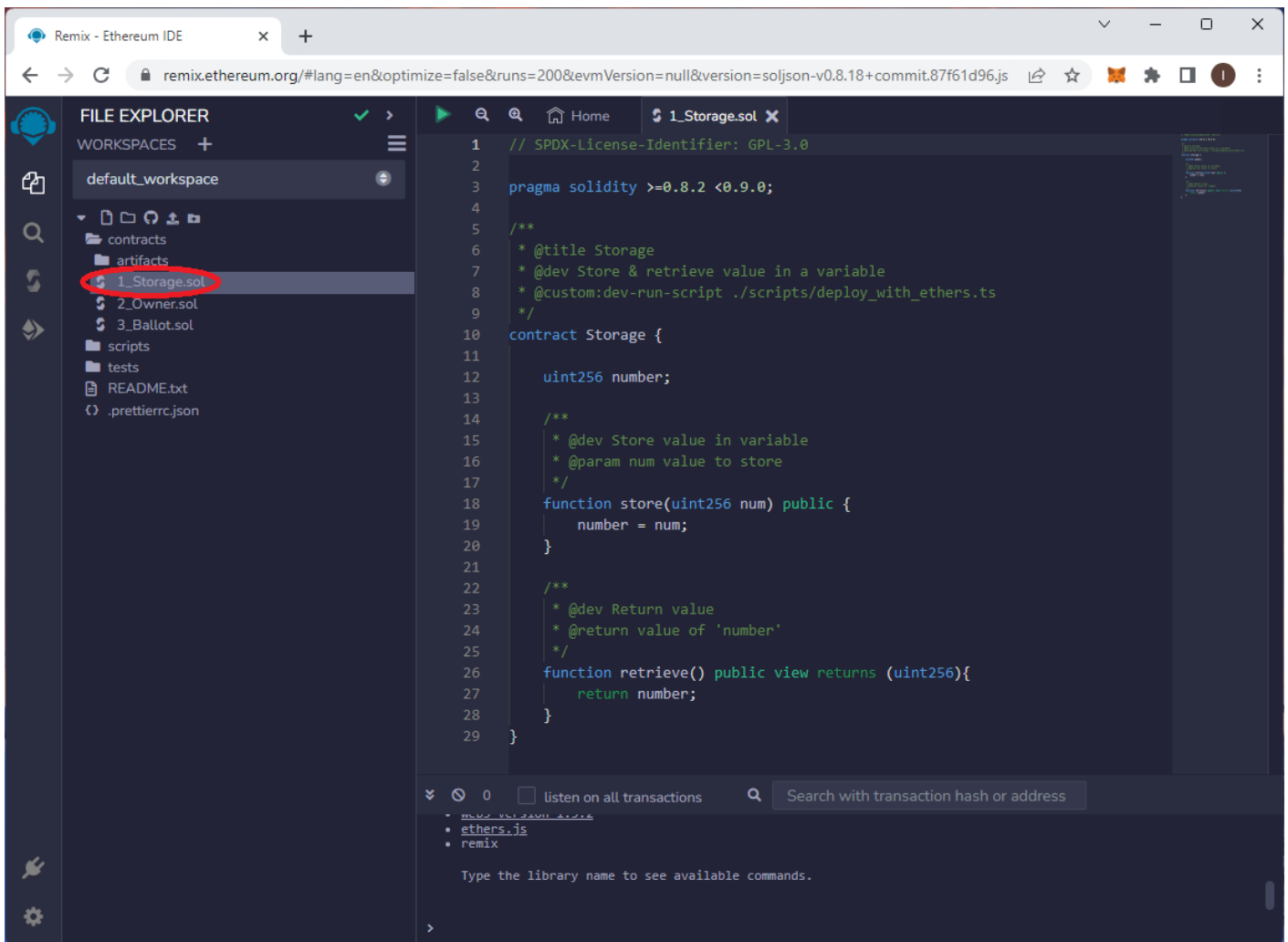


# Deploying a smart contract on BPX mainnet using Remix IDE

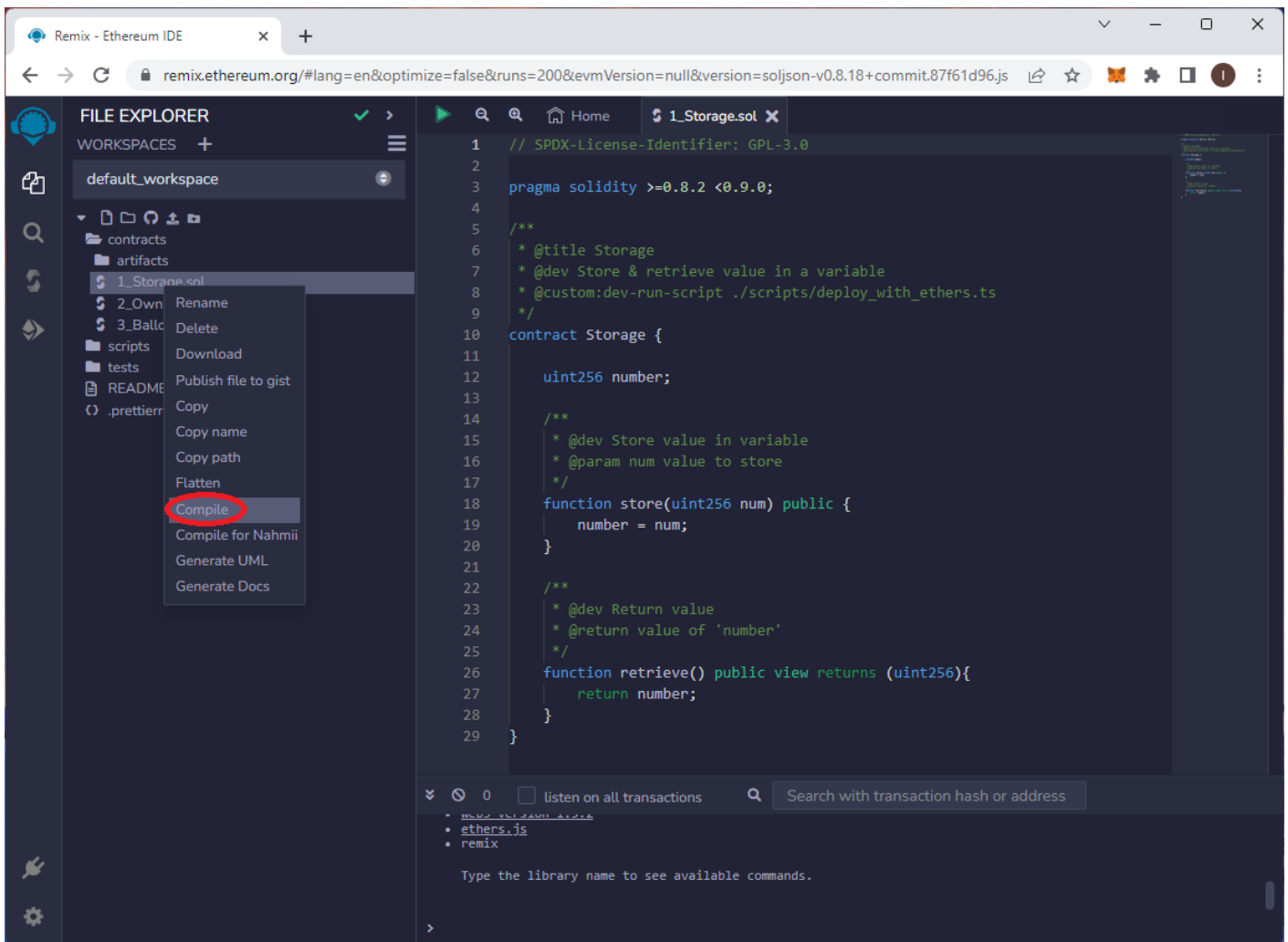
1. Open [Remix IDE](#).



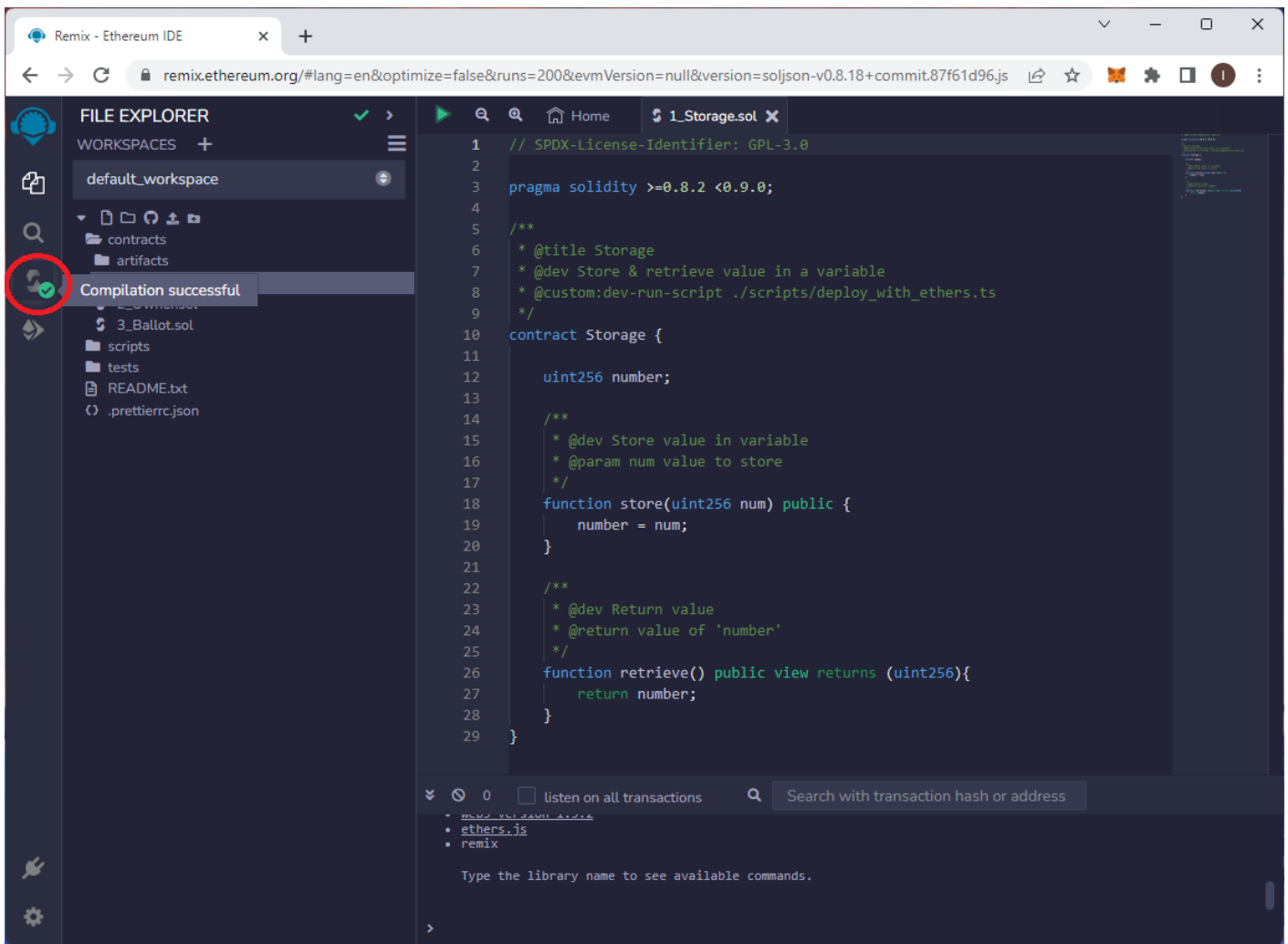
2. In our example, we will use one of the sample contracts from the IDE. This contract provides two functions: the first, `store`, allows you to save any number in the contract, while the second, `retrieve`, enables you to read the stored number. Open the `contracts/1_Storage.sol` file.



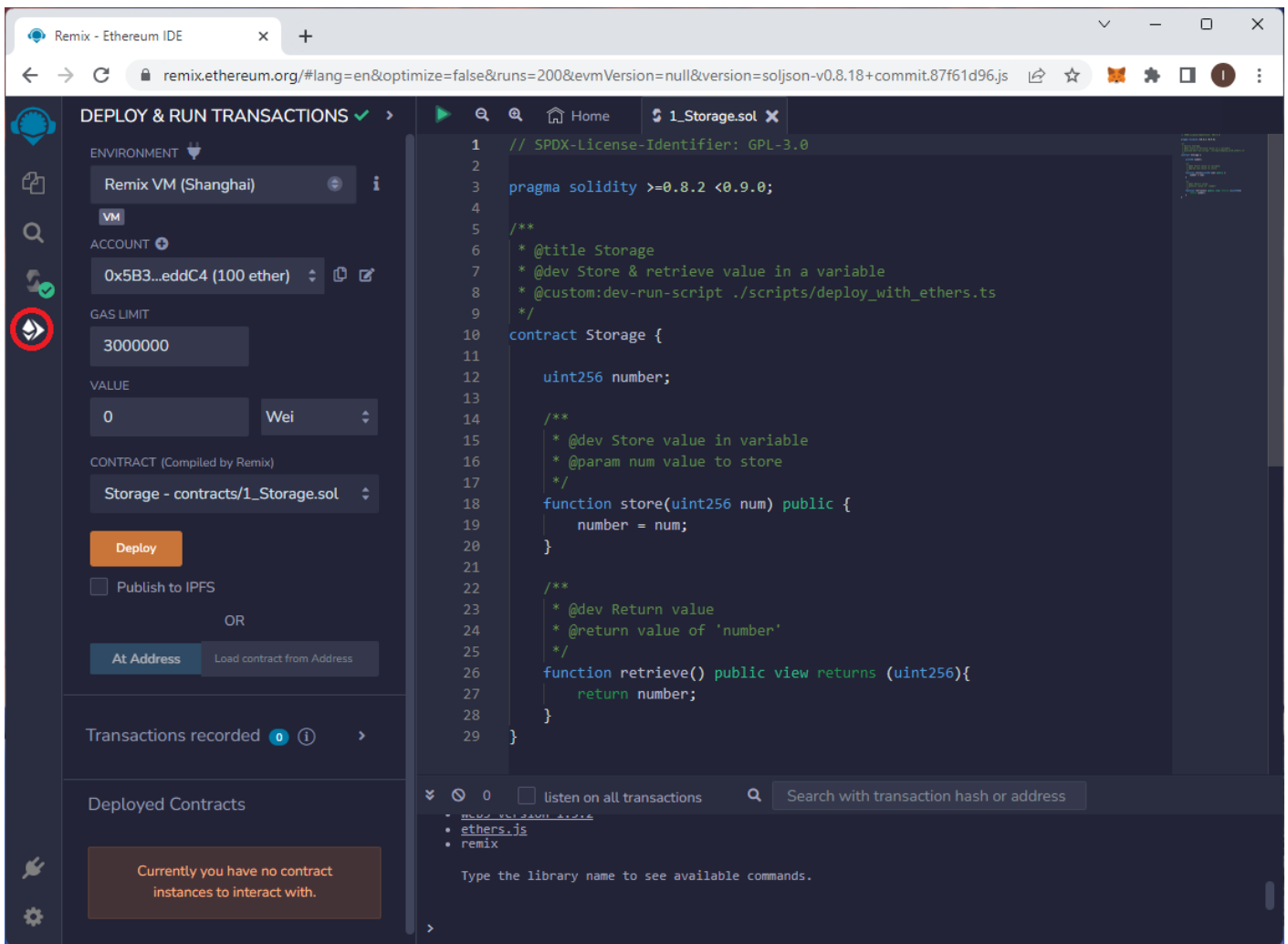
3. Compile the contract by right-clicking on the file name and selecting **Compile**.



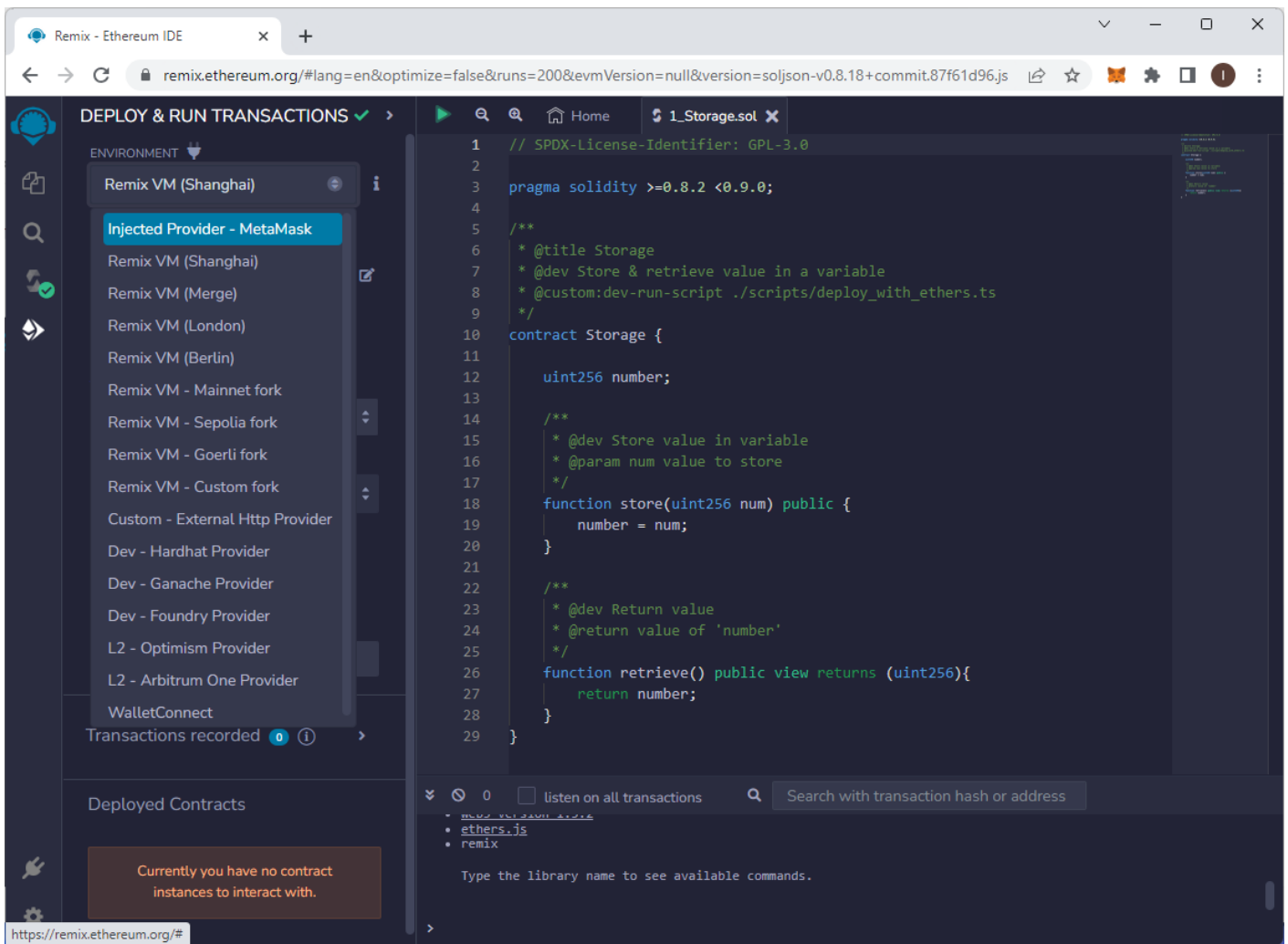
4. When the source code is error-free and the compilation is successful, you will see a green success icon.



5. Go to the **Deploy** tab.



6. Expand the **Environment** list and select **Injected Provider - MetaMask**.



7. Now MetaMask will prompt you for permission to connect to Remix IDE. Agree to the connection.

## Connect with MetaMask

Select the account(s) to use on this site

[New account](#)



Account 1 (0xfdb...cd...  
1240279.9999685 BPX



Only connect with sites you trust. [Learn more](#)

Cancel

Next

## Connect to Account 1 (0xfdb...cda3)

Allow this site to:



See address, account balance, activity  
and suggest transactions to approve

Only connect with sites you trust. [Learn more](#)

Cancel

Connect

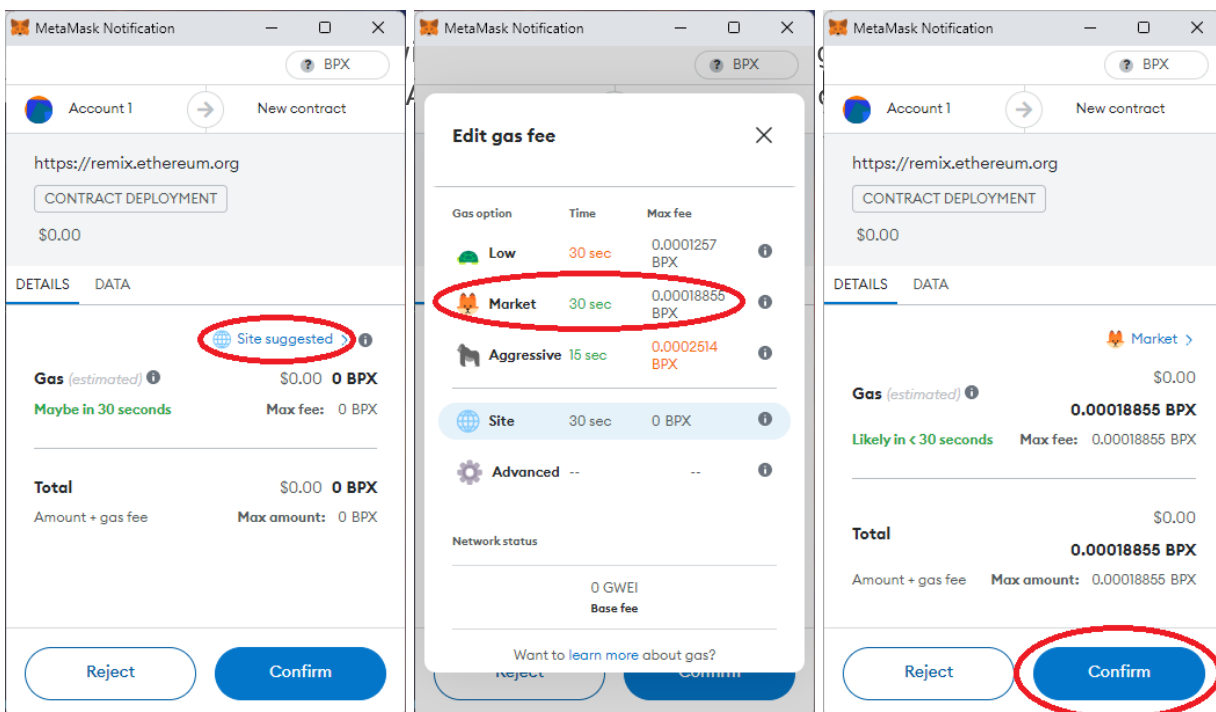
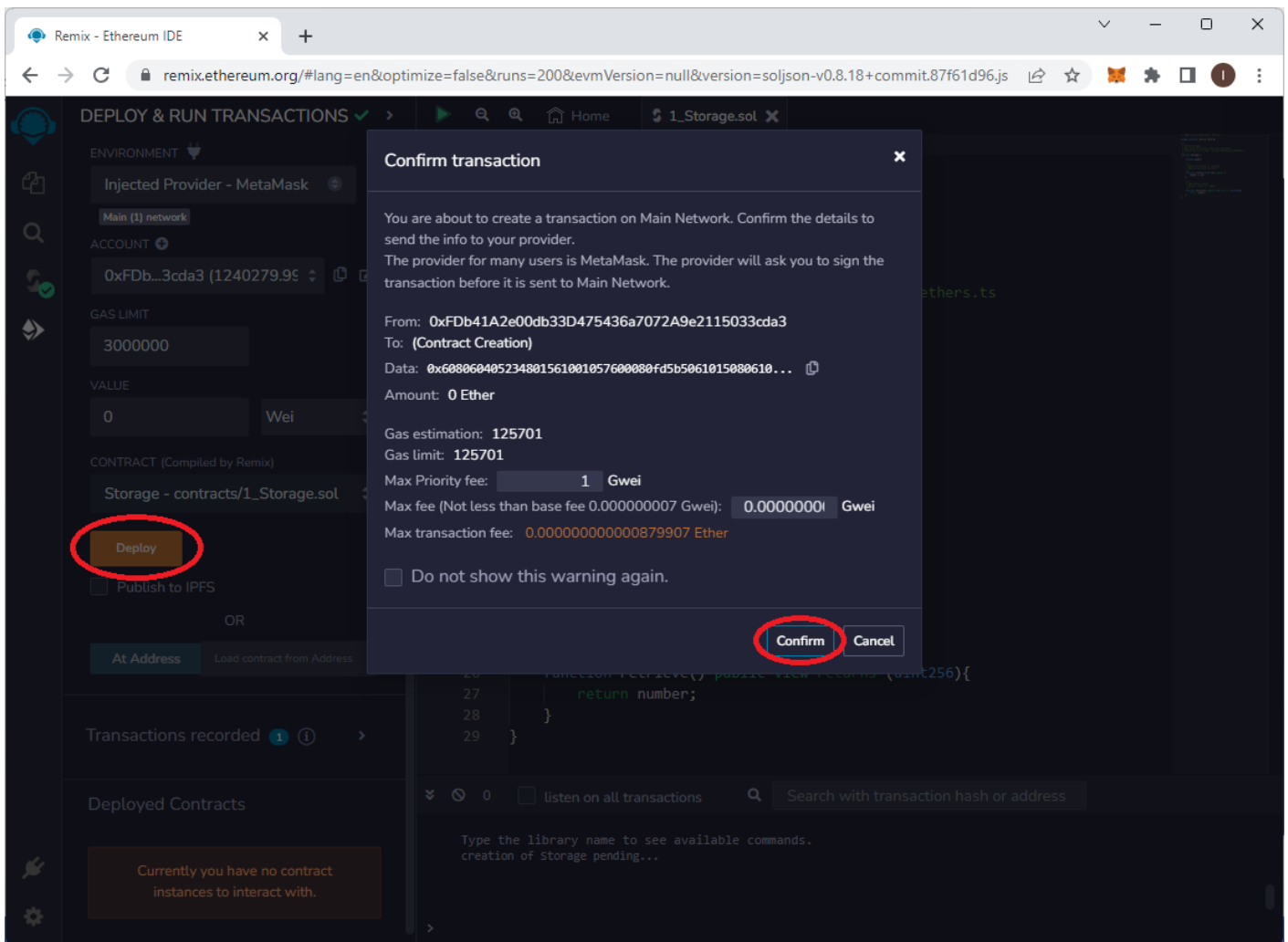
8. After successfully connecting, you should see your account address and its balance in the highlighted field.

The screenshot displays the Remix Ethereum IDE interface. On the left sidebar, the 'DEPLOY & RUN TRANSACTIONS' panel is active. It shows the 'ENVIRONMENT' set to 'Injected Provider - MetaMask', the 'ACCOUNT' as '0xFDb...3cda3 (1240279.9%)', and the 'GAS LIMIT' set to '3000000'. The 'VALUE' is '0' and the unit is 'Wei'. The 'CONTRACT' is 'Storage - contracts/1\_Storage.sol'. A red circle highlights the account address '0xFDb...3cda3 (1240279.9%)'. Below these settings are buttons for 'Deploy', 'Publish to IPFS', and 'At Address'. The main editor area shows the Solidity code for the 'Storage' contract, which includes a 'store' function and a 'retrieve' function. The bottom panel shows the 'Deployed Contracts' section with a message: 'Currently you have no contract instances to interact with.'

```
1 // SPDX-License-Identifier: GPL-3.0
2
3 pragma solidity >=0.8.2 <0.9.0;
4
5 /**
6  * @title Storage
7  * @dev Store & retrieve value in a variable
8  * @custom:dev-run-script ./scripts/deploy_with_ethers.ts
9  */
10 contract Storage {
11
12     uint256 number;
13
14     /**
15      * @dev Store value in variable
16      * @param num value to store
17      */
18     function store(uint256 num) public {
19         number = num;
20     }
21
22     /**
23      * @dev Return value
24      * @return value of 'number'
25      */
26     function retrieve() public view returns (uint256){
27         return number;
28     }
29 }
```

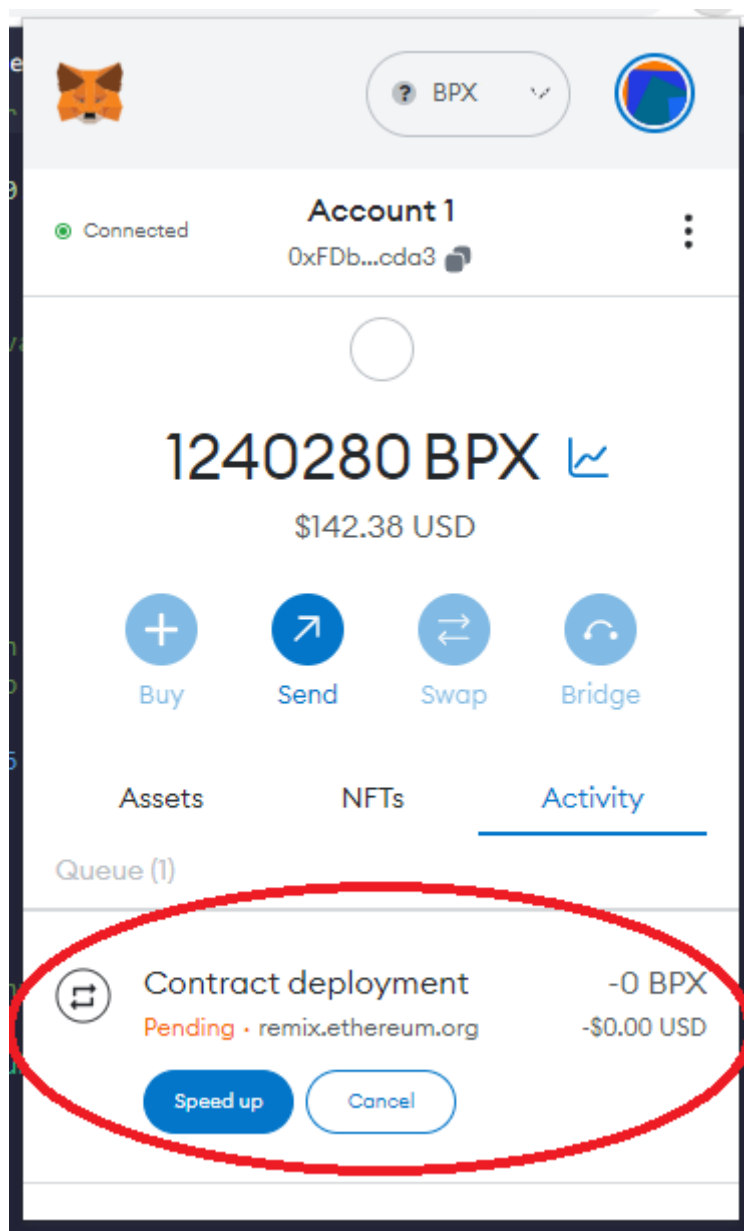
9. Click "**Deploy**" to begin deploying your contract. Confirm the suggested gas settings.





ested," then

11. Your contract is now deploying. If you open the MetaMask window, you should see a new pending transaction.



12. Once the transaction is confirmed by the blockchain, you will see its confirmation and a new entry in the Deployed Contracts section. Your contract has been successfully deployed.

The screenshot displays the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is visible, showing the environment set to 'Injected Provider - MetaMask' on the 'Main (1) network'. The account address is '0xFDb...3cda3 (1240279.99%)'. The gas limit is set to '3000000' and the value is '0 Wei'. The contract selected for deployment is 'Storage - contracts/1\_Storage.sol'. The 'Deploy' button is highlighted. Below the deployment settings, the 'Transactions recorded' section shows one transaction. The 'Deployed Contracts' section lists 'STORAGE AT 0X5CD...0E9B3 (BLOC)' with a copy icon and a close button. The main editor displays the Solidity code for the 'Storage' contract, which includes a 'store' function and a 'retrieve' function. The bottom panel shows the transaction log, indicating a successful deployment at block 27514, transaction index 0, from the account 0xFDb...3cda3 to the Storage constructor, with a value of 0 wei. The transaction hash is 0x570...4e2d4. A 'Debug' button is also present in the transaction log.

13. You can use the highlighted button to copy the address of your new contract.

The image shows the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is visible, showing the environment (Injected Provider - MetaMask), account (0xFDb...3cda3), gas limit (3000000), and value (0 Wei). The contract selected is 'Storage - contracts/1\_Storage.sol'. The 'Deploy' button is highlighted. Below the sidebar, the 'Deployed Contracts' section shows a list of contracts, with 'STORAGE AT 0X5CD...0E9B3 (BLOC)' selected and its icon circled in red. The main editor displays the Solidity code for the 'Storage' contract, which includes a 'store' function and a 'retrieve' function. The bottom panel shows the transaction details for the deployment, including the block number (27514), transaction index (0), and the contract address (0x570...4e2d4). A 'Debug' button is visible next to the transaction details.

```
1 // SPDX-License-Identifier: GPL-3.0
2
3 pragma solidity >=0.8.2 <0.9.0;
4
5 /**
6  * @title Storage
7  * @dev Store & retrieve value in a variable
8  * @custom:dev-run-script ./scripts/deploy_with_ethers.ts
9  */
10 contract Storage {
11
12     uint256 number;
13
14     /**
15      * @dev Store value in variable
16      * @param num value to store
17      */
18     function store(uint256 num) public {
19         number = num;
20     }
21
22     /**
23      * @dev Return value
24      * @return value of 'number'
25      */
26     function retrieve() public view returns (uint256){
27         return number;
28     }
29 }
```

Transaction details: [block:27514 txIndex:0] from: 0xFDb...3cda3 to: Storage.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0x570...4e2d4

14. You can interact with your contract directly from the IDE. Expand the list of contract methods to access them.

The screenshot displays the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is active. It shows a 'VALUE' of 0 Wei and a 'CONTRACT' dropdown set to 'Storage - contracts/1\_Storage.sol'. The 'Deploy' button is highlighted. Below it, there are options to 'Publish to IPFS' or 'At Address'. The 'Transactions recorded' section shows one transaction. The 'Deployed Contracts' section lists a contract 'STORAGE AT 0x5CD...0E9B3 (BLK)' with a balance of 0 ETH. A red circle highlights a dropdown arrow next to the contract name. Below the contract name, there are 'store' and 'retrieve' buttons. The 'store' button is highlighted. The 'Low level interactions' section shows a 'CALLDATA' field and a 'Transact' button. The main editor area displays the Solidity code for the 'Storage' contract. The code includes a pragma statement, a title comment, a dev comment, a custom script path, and two functions: 'store' and 'retrieve'. The 'store' function takes a 'uint256 num' and sets 'number = num'. The 'retrieve' function returns the 'number'. The bottom status bar shows a green checkmark, indicating a successful transaction. It displays the transaction details: '[block:27514 txIndex:0] from: 0xFDb...3cda3 to: Storage.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0x570...4e2d4'. A 'Debug' button is also visible.

```
1 // SPDX-License-Identifier: GPL-3.0
2
3 pragma solidity >=0.8.2 <0.9.0;
4
5 /**
6  * @title Storage
7  * @dev Store & retrieve value in a variable
8  * @custom:dev-run-script ./scripts/deploy_with_ethers.ts
9  */
10 contract Storage {
11
12     uint256 number;
13
14     /**
15      * @dev Store value in variable
16      * @param num value to store
17      */
18     function store(uint256 num) public {
19         number = num;
20     }
21
22     /**
23      * @dev Return value
24      * @return value of 'number'
25      */
26     function retrieve() public view returns (uint256){
27         return number;
28     }
29 }
```

15. Let's call the first function (`store`) to save a number in our sample contract. Enter a random number in the text field next to the "**store**" button, and then press the "**store**" button.

The screenshot displays the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is active. It shows a 'VALUE' of 0 Wei, a 'CONTRACT' dropdown set to 'Storage - contracts/1\_Storage.sol', and a 'Deploy' button. Below this, there's a section for 'Deployed Contracts' showing a contract named 'STORAGE AT 0x5CD...0E9B3 (BLK)' with a balance of 0 ETH. A red circle highlights the 'store' button next to the value '150'. Below that is a 'retrieve' button. At the bottom of the sidebar is a 'Low level interactions' section with a 'Transact' button. The main editor area shows the Solidity code for the 'Storage' contract, which includes a 'store' function and a 'retrieve' function. The bottom status bar shows a transaction confirmation: '[block:27521 txIndex:0] from: 0xFDb...3cda3 to: Storage.store(uint256) 0x5CD...0E9B3 value: 0 wei data: 0x605...00096 logs: 0 hash: 0x8a5...01de0'.

16. Confirm the transaction in your wallet as you did when deploying the contract. Saving data to a smart contract requires a transaction on the blockchain.

17. When the transaction is confirmed, call the second method (`retrieve`) to read the number stored in the smart contract. Click the "**retrieve**" button, and the blockchain will return the value stored in the contract.

The screenshot displays the Remix Ethereum IDE interface. The top bar shows the browser address: `remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.18+commit.87f61d96.js`. The left sidebar contains a 'DEPLOY & RUN TRANSACTIONS' panel with a 'VALUE' input set to 0 Wei, a 'CONTRACT' dropdown showing 'Storage - contracts/1\_Storage.sol', and a 'Deploy' button. Below this, there's a 'Transactions recorded' section with 2 transactions, and a 'Deployed Contracts' section showing a contract at address 0x5CD...0E9B3 (BLK) with a balance of 0 ETH. The 'store' button is highlighted in orange, and the 'retrieve' button is circled in red. The 'Low level interactions' section shows a 'CALLDATA' input field and a 'Transact' button. The main editor area displays the Solidity code for the 'Storage' contract, which includes a 'store' function and a 'retrieve' function. The bottom panel shows a debug console with a call from address 0xFDb41A2e00db330475436a7072A9e2115033cda3 to the 'Storage.retrieve()' function, returning data 0x2e6...4cec1.

```
1 // SPDX-License-Identifier: GPL-3.0
2
3 pragma solidity >=0.8.2 <0.9.0;
4
5 /**
6  * @title Storage
7  * @dev Store & retrieve value in a variable
8  * @custom:dev-run-script ./scripts/deploy_with_ethers.ts
9  */
10 contract Storage {
11
12     uint256 number;
13
14     /**
15      * @dev Store value in variable
16      * @param num value to store
17      */
18     function store(uint256 num) public {
19         number = num;
20     }
21
22     /**
23      * @dev Return value
24      * @return value of 'number'
25      */
26     function retrieve() public view returns (uint256){
27         return number;
28     }
29 }
```

Revision #3

Created 8 June 2023 07:45:26 by Admin

Updated 5 November 2024 12:05:12 by Admin