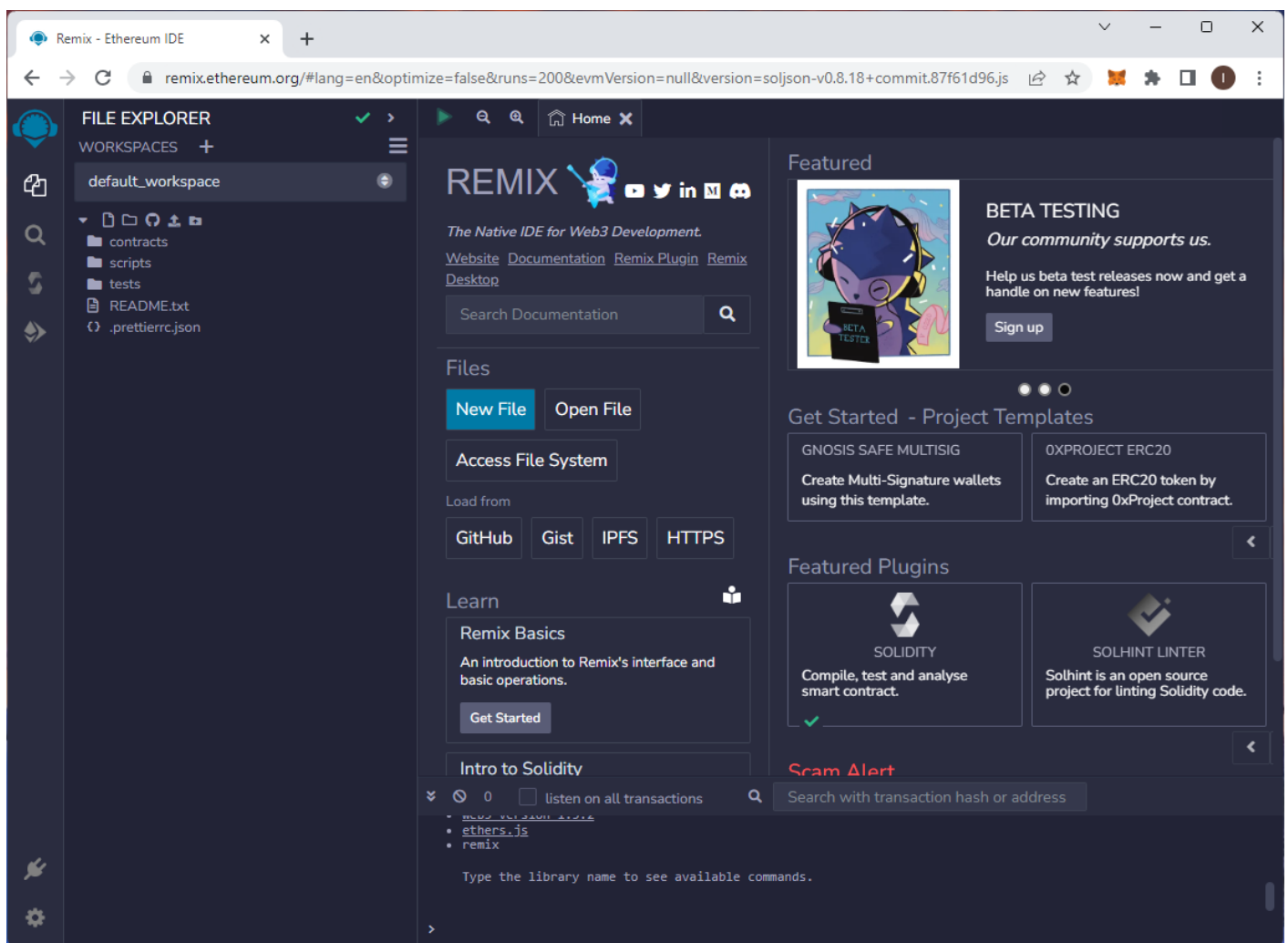
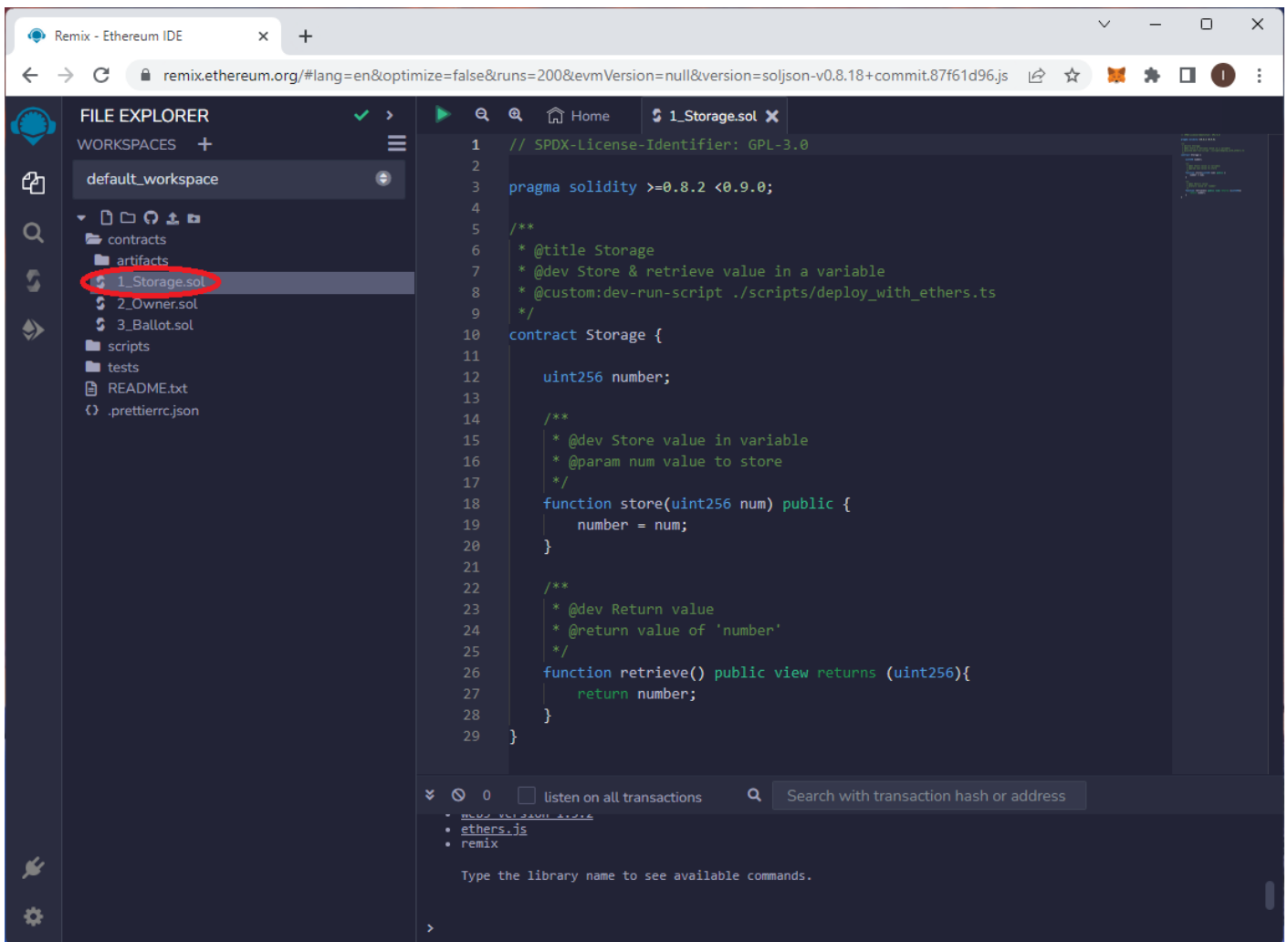


# Deploying a smart contract on BPX mainnet using Remix IDE

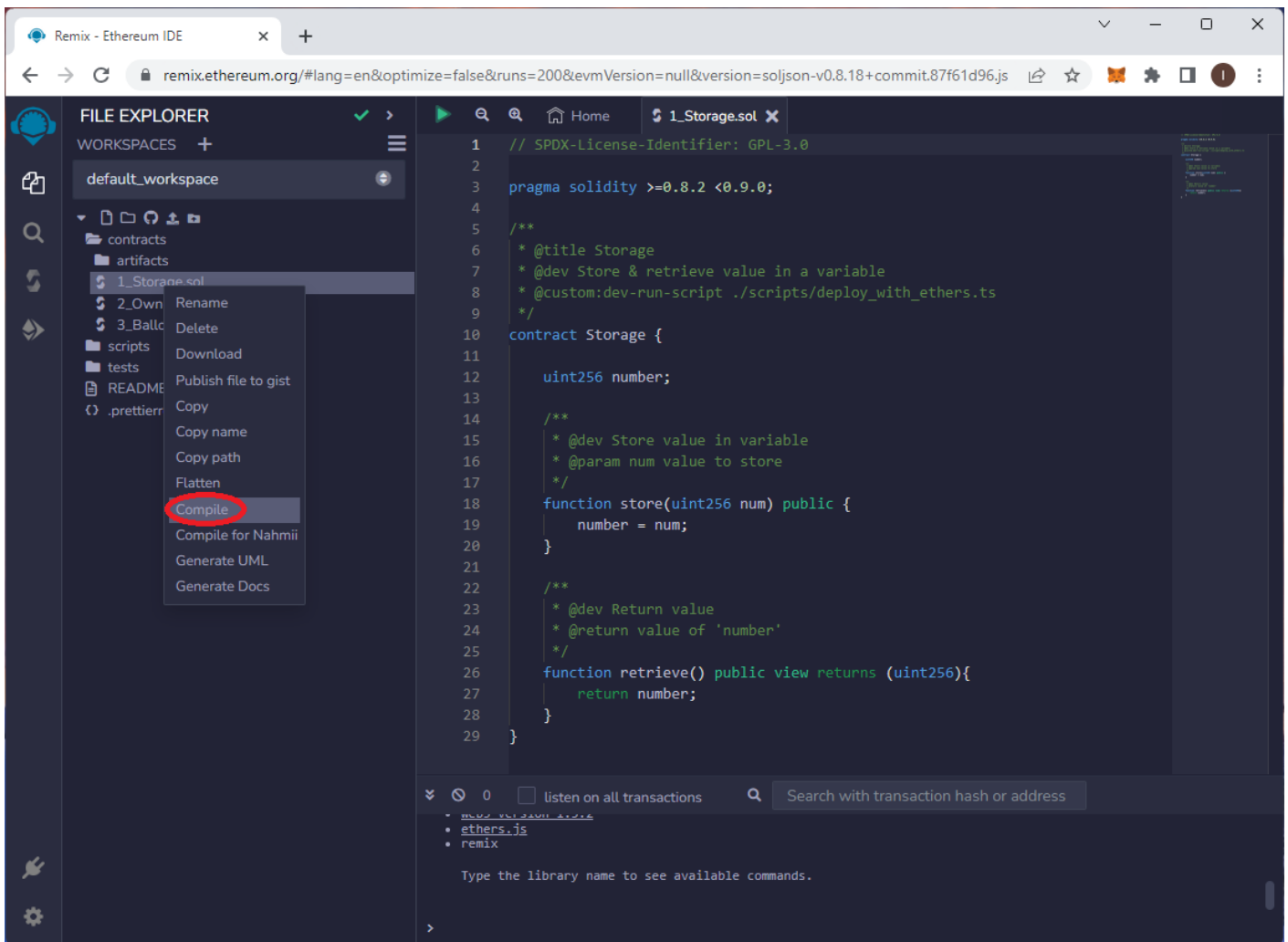
1. Open [Remix IDE](#).



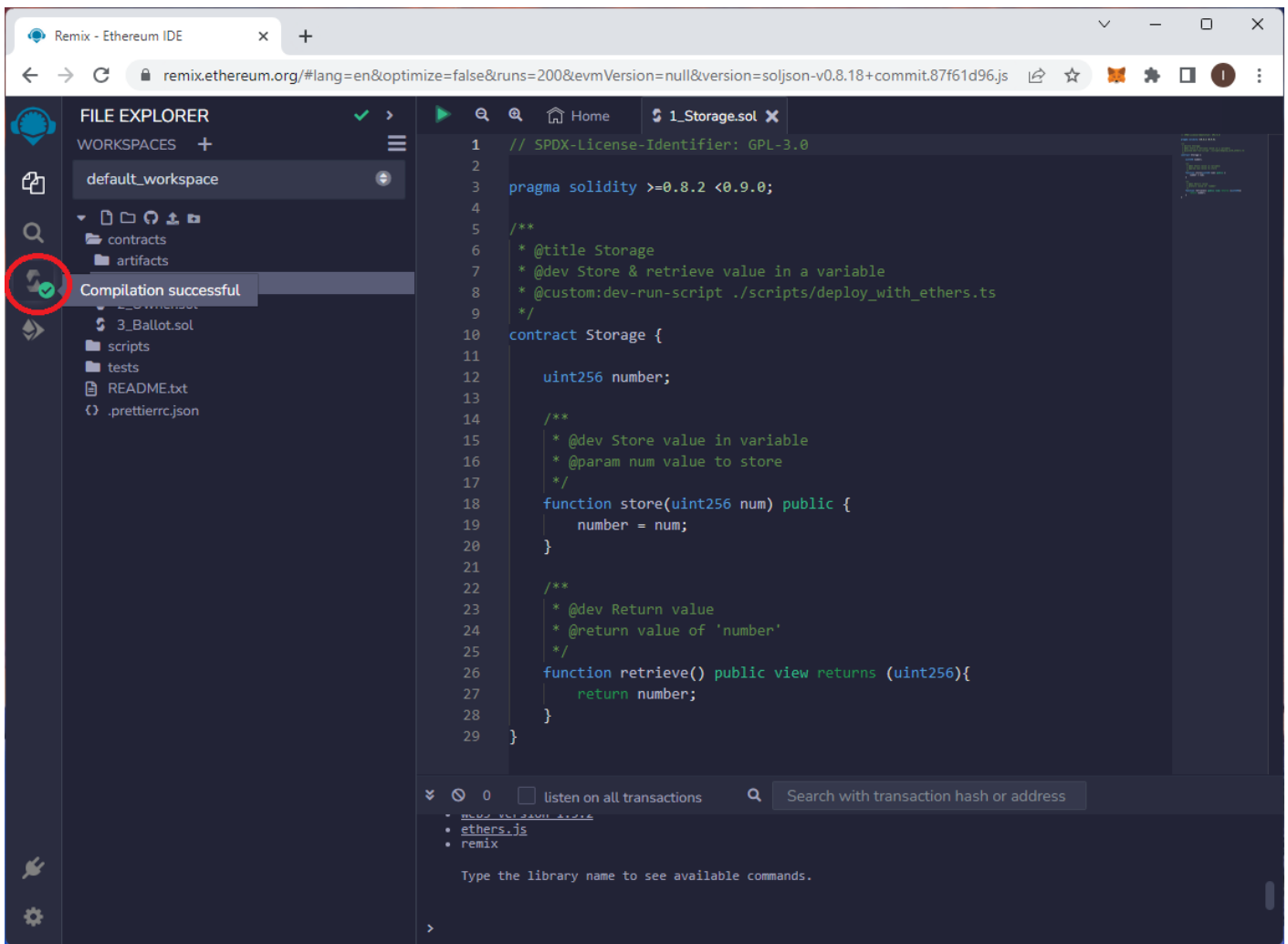
2. In our example, we will use one of the sample contracts from the IDE. This contract provides two functions: the first, `store`, allows you to save any number in the contract, while the second, `retrieve`, enables you to read the stored number. Open the `contracts/1_Storage.sol` file.



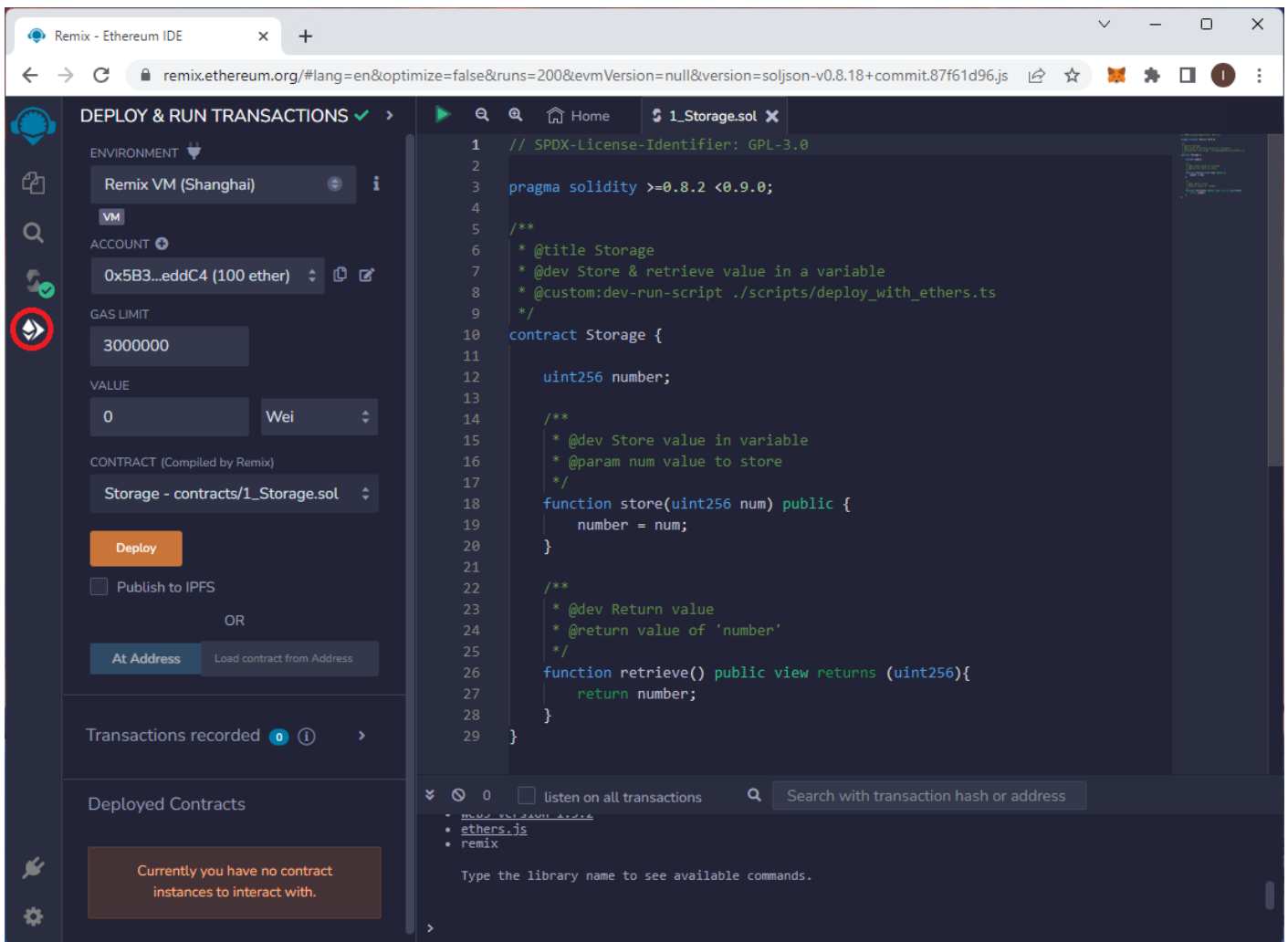
3. Compile the contract by right-clicking on the file name and selecting **Compile**.



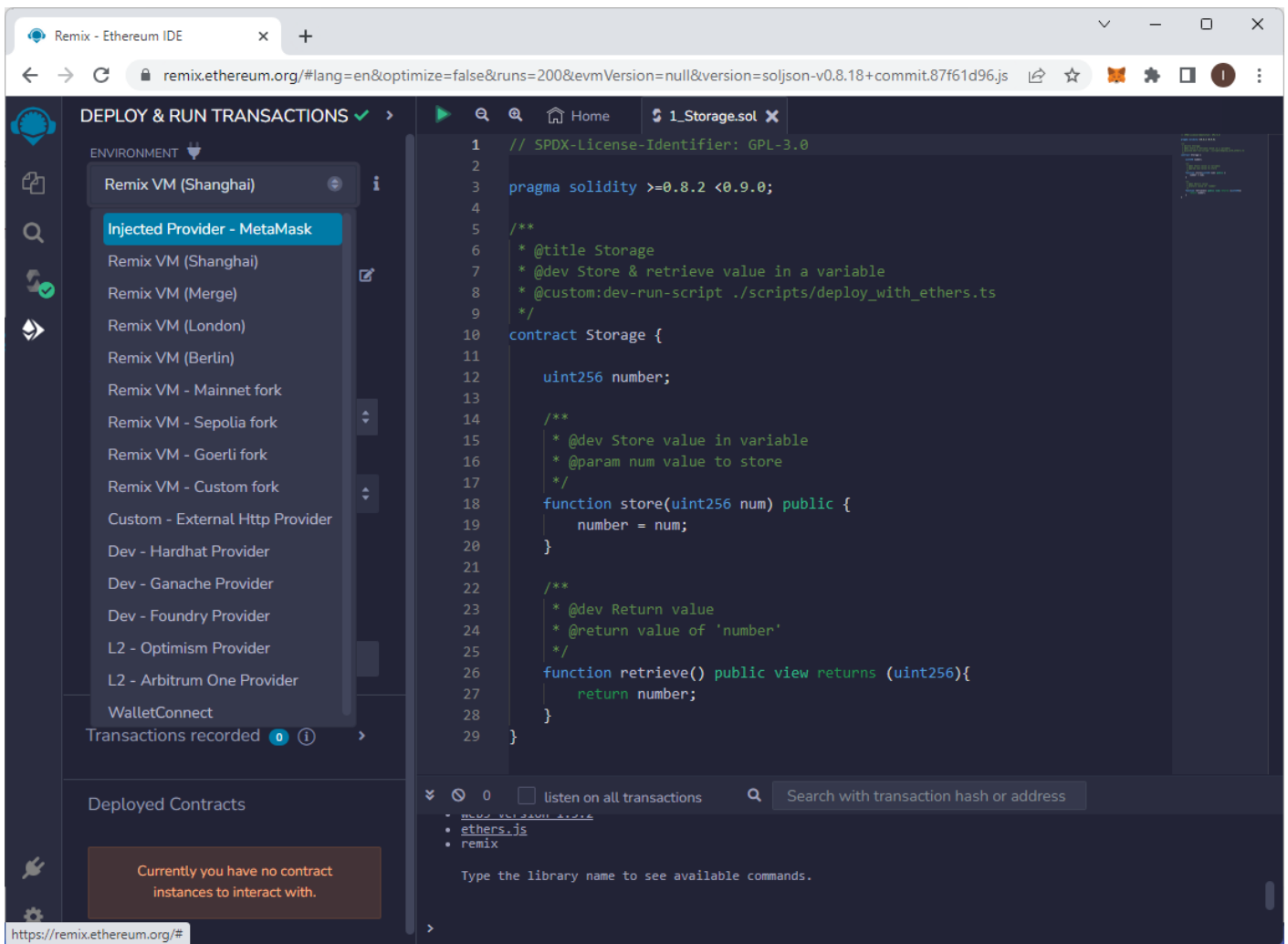
4. When the source code is error-free and the compilation is successful, you will see a green success icon.



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

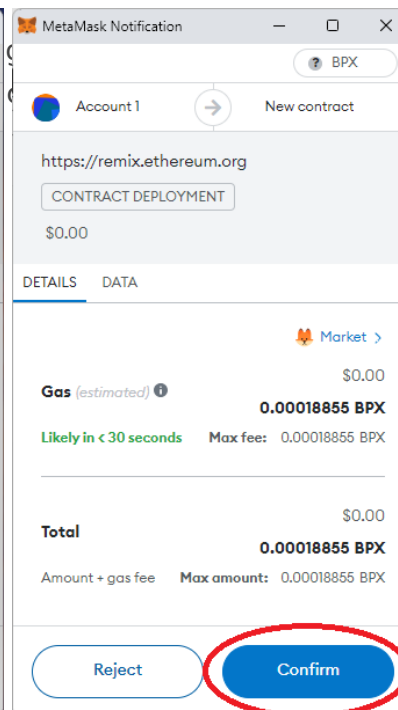
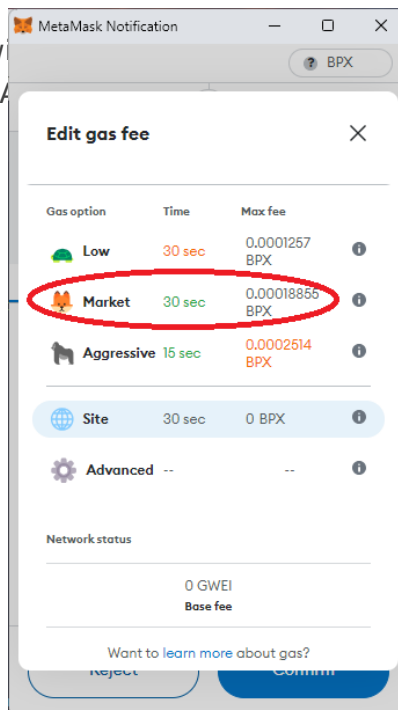
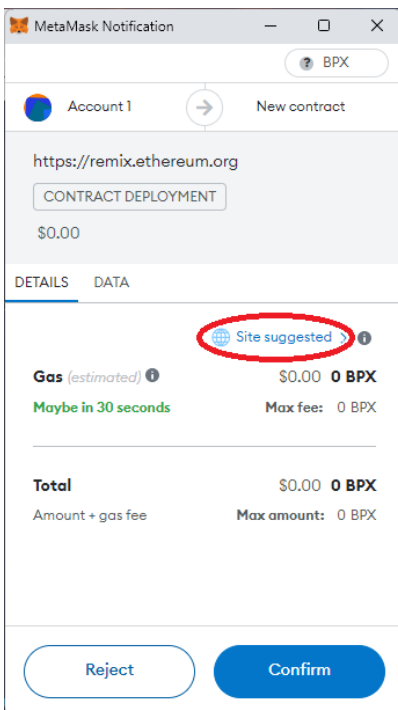
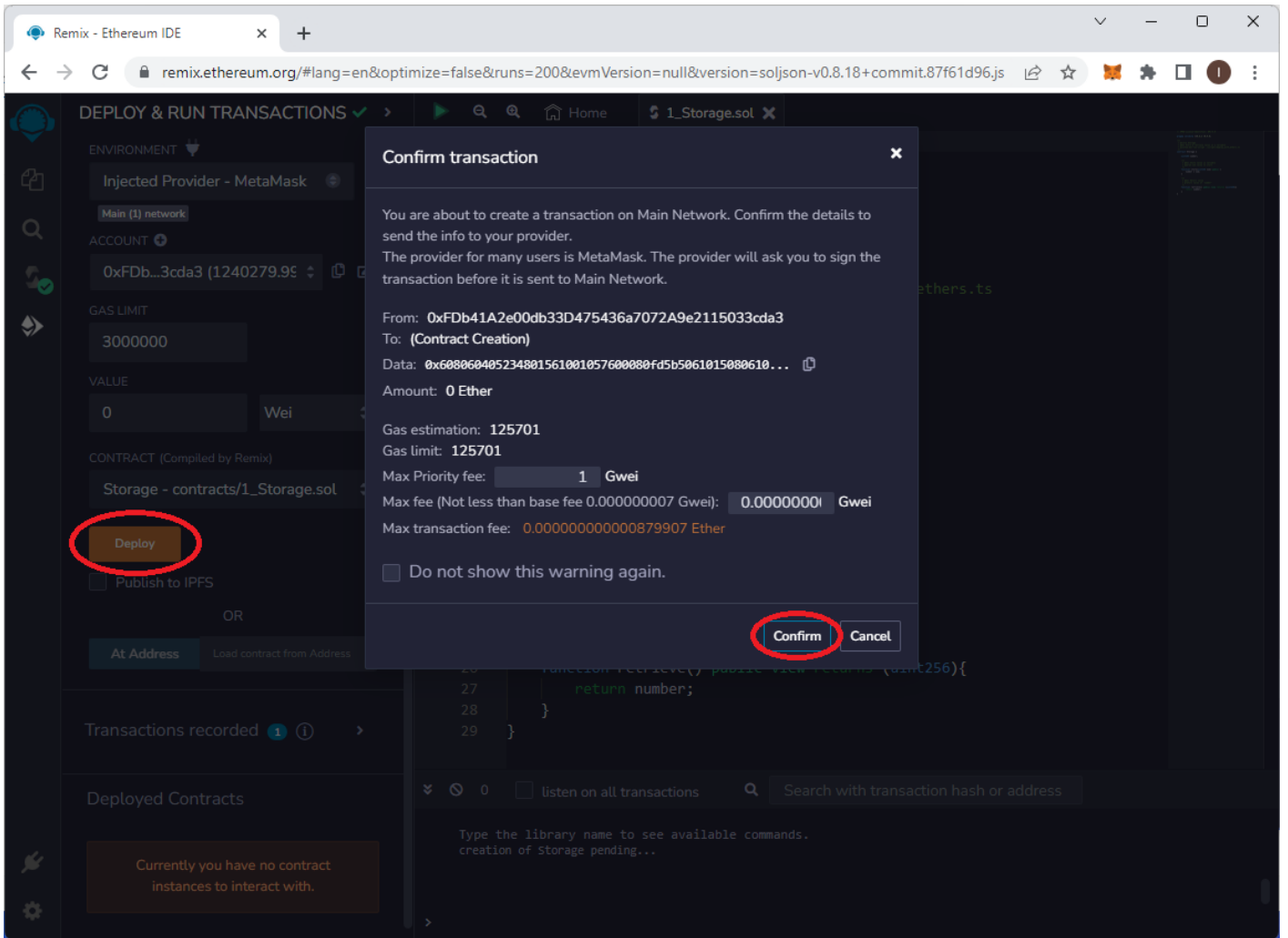
8. After successfully connecting, you should see your account address and its balance in the highlighted field.

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 }
```

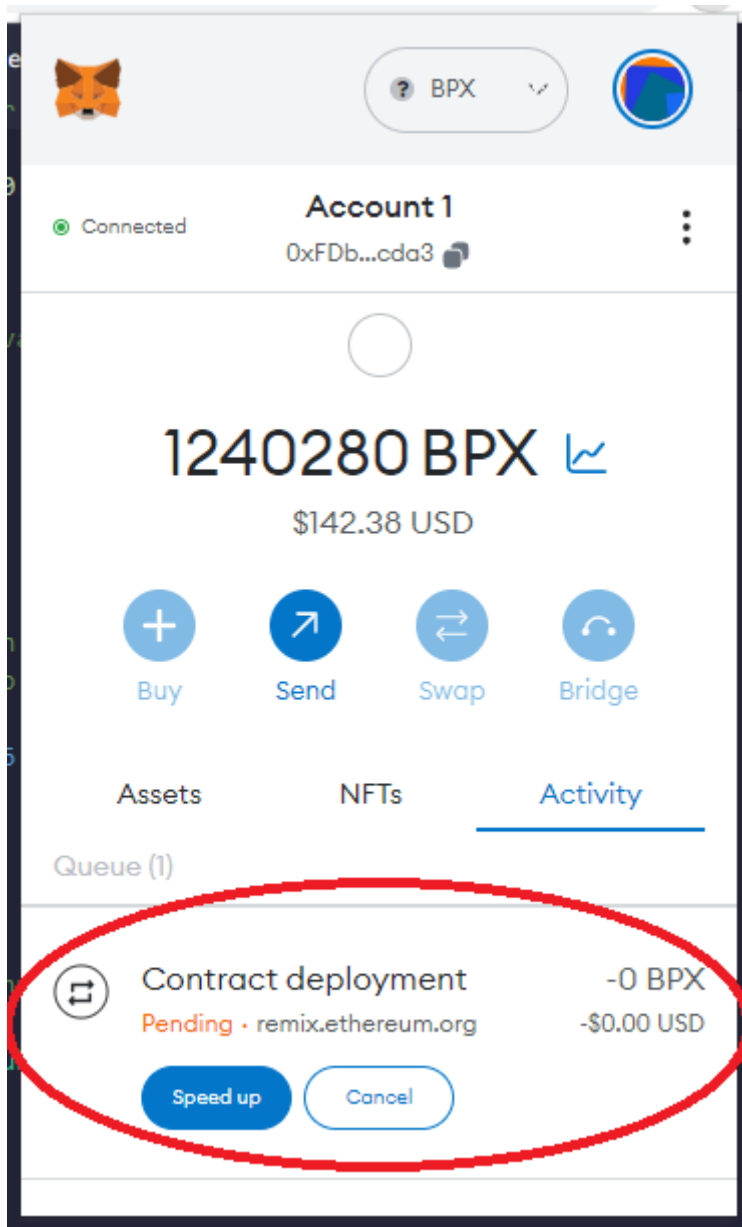
9. Click "**Deploy**" to begin deploying your contract. Confirm the suggested gas settings.





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.

Remix - Ethereum IDE

remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.18+commit.87f61d96.js

### DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Injected Provider - MetaMask

ACCOUNT: 0xFDb...3cda3 (1240279.99%)

GAS LIMIT: 3000000

VALUE: 0 Wei

CONTRACT (Compiled by Remix): Storage - contracts/1\_Storage.sol

Deploy

☐ Publish to IPFS

OR

At Address: Load contract from Address

Transactions recorded: 1

#### Deployed Contracts

- STORAGE AT 0X5CD...0E9B3 (BLOC)

```
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 }
```

view on etherscan

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

Debug

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

The screenshot displays the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is active, showing the 'ENVIRONMENT' set to 'Injected Provider - MetaMask', the 'ACCOUNT' as '0xFDb...3cda3 (1240279.99\$)', and the 'GAS LIMIT' set to '3000000'. The 'VALUE' is '0' in 'Wei'. The 'CONTRACT' is 'Storage - contracts/1\_Storage.sol'. A 'Deploy' button is visible, along with options to 'Publish to IPFS' or 'Load contract from Address'. Below this, it shows 'Transactions recorded' and 'Deployed Contracts'. In the 'Deployed Contracts' section, a contract is listed as 'STORAGE AT 0X5CD...0E9B3 (BLOC)' with a red circle highlighting an icon next to it. The main editor shows the Solidity code for '1\_Storage.sol', which includes a pragma statement, comments, and two functions: 'store' and 'retrieve'. The bottom panel shows transaction details for '[block:27514 txIndex:0] from: 0xFDb...3cda3 to: Storage.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0x570...4e2d4', with a 'Debug' button.

```
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 }
```

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, showing a 'VALUE' of 0 Wei and a 'CONTRACT' dropdown set to 'Storage - contracts/1\_Storage.sol'. Below this, there are buttons for 'Deploy', 'Publish to IPFS', and 'At Address'. The 'Deployed Contracts' section shows a contract named 'STORAGE AT 0x5CD...0E9B3 (BLK)' with a balance of 0 ETH. It includes 'store' and 'retrieve' buttons, with a text input field next to the 'store' button containing 'uint256 num'. The 'Low level interactions' section at the bottom has a 'Transact' button. The main editor on the right shows the Solidity code for the 'Storage' contract, which includes a 'store' function and a 'retrieve' function. The bottom status bar shows a successful transaction: '[block:27514 txIndex:0] from: 0xFDb...3cda3 to: Storage.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0x570...4e2d4'.

```
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 the deployed contracts, there's a 'Low level interactions' section with a 'CALLDATA' input field and 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 and 'Wei' as the unit. Below this, the 'CONTRACT' dropdown is set to 'Storage - contracts/1\_Storage.sol'. A 'Deploy' button is visible, along with a 'Publish to IPFS' checkbox. The 'Transactions recorded' section shows 2 transactions. The 'Deployed Contracts' section lists a contract at address 0x5CD...0E9B3 (BLK) with a balance of 0 ETH. It features two buttons: 'store' and 'retrieve'. The 'retrieve' button is circled in red, and below it, the output '0: uint256: 150' is displayed. 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 transaction log with a call from address 0xFDb41A2e00db330475436a7072A9e2115033cda3 to 'Storage.retrieve()' with data '0x2e6...4cec1'. A 'Debug' button is present next to the transaction log.

```
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