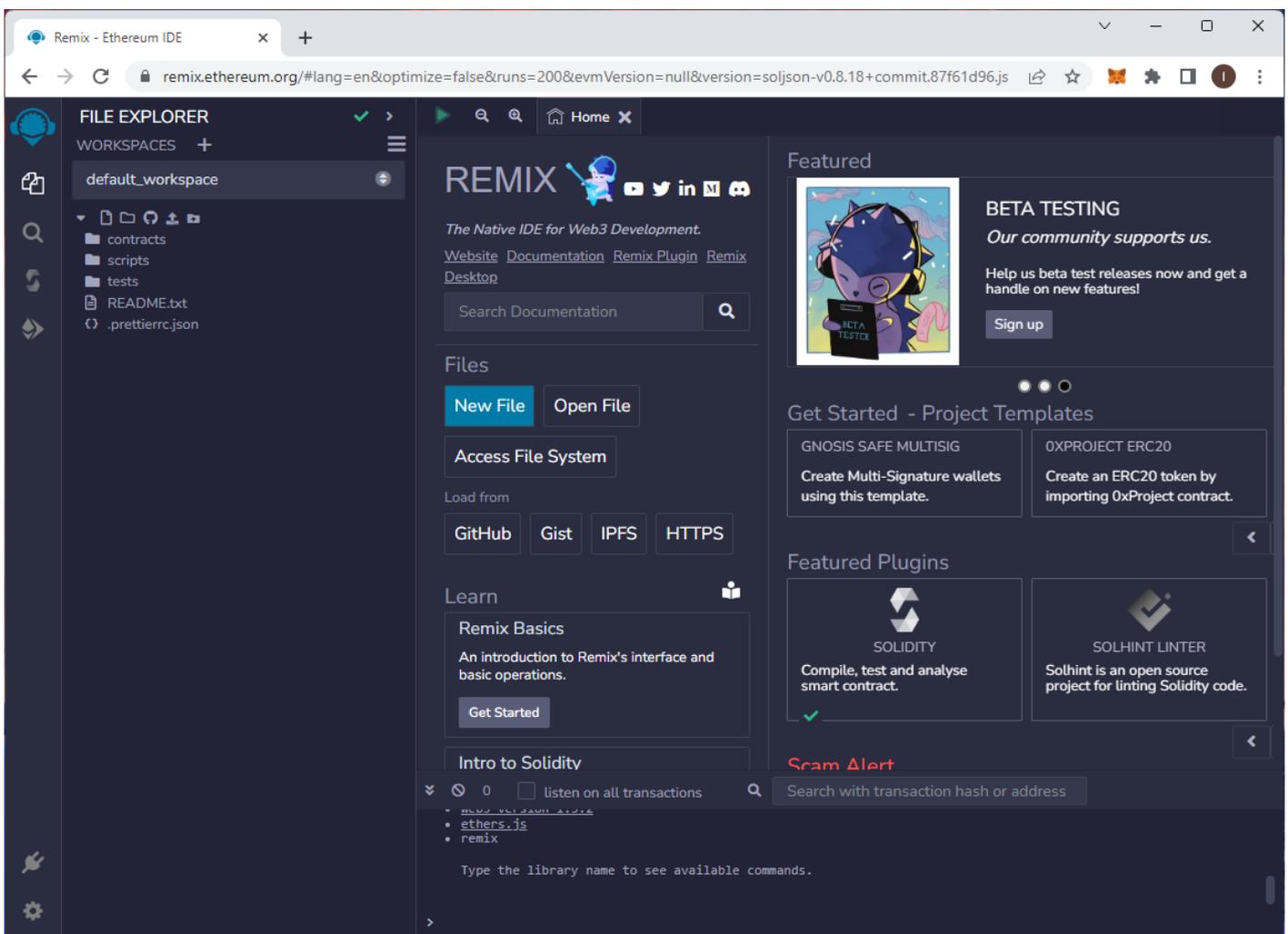
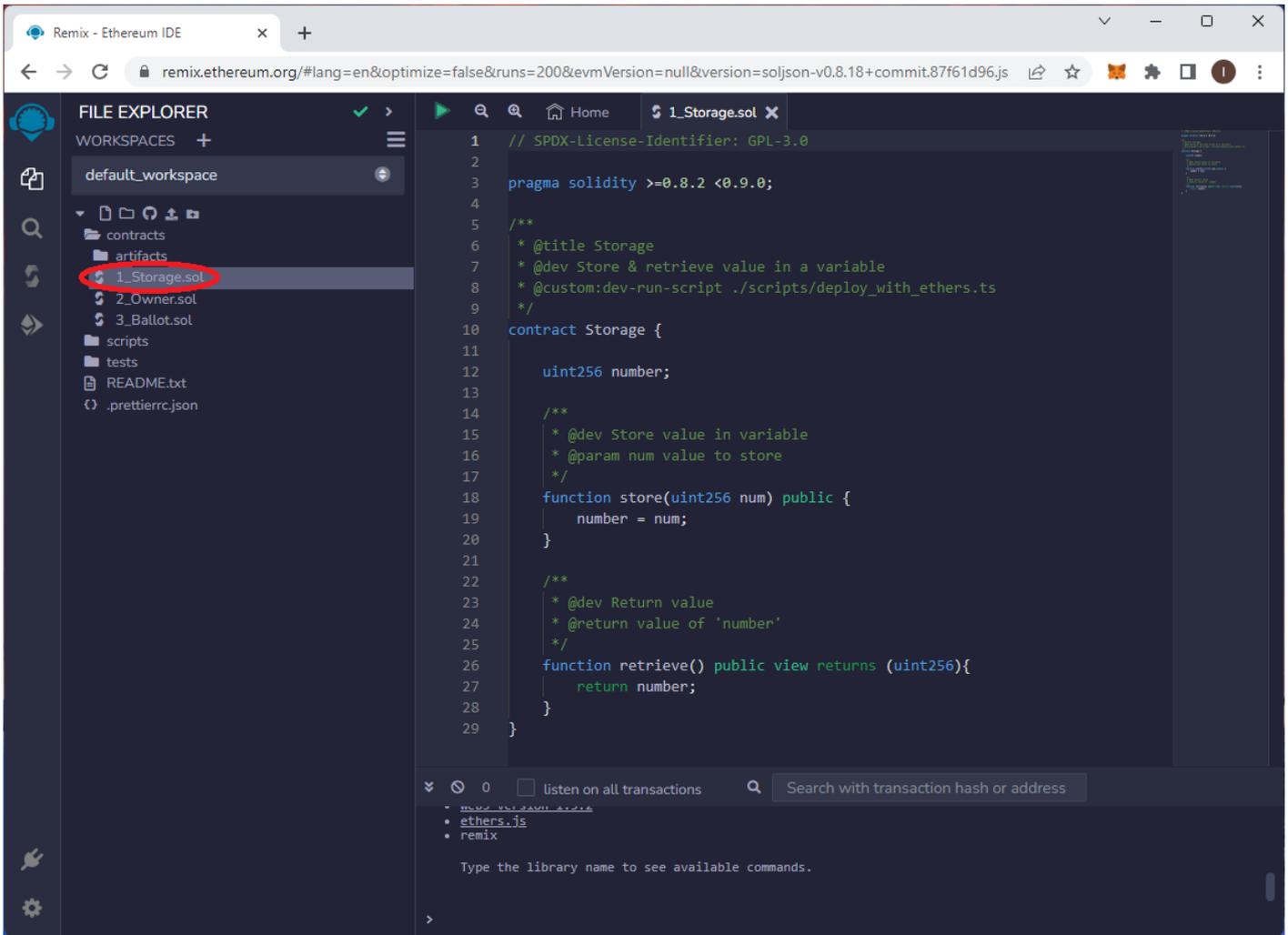


Deploying a smart contract on BPX mainnet using Remix IDE

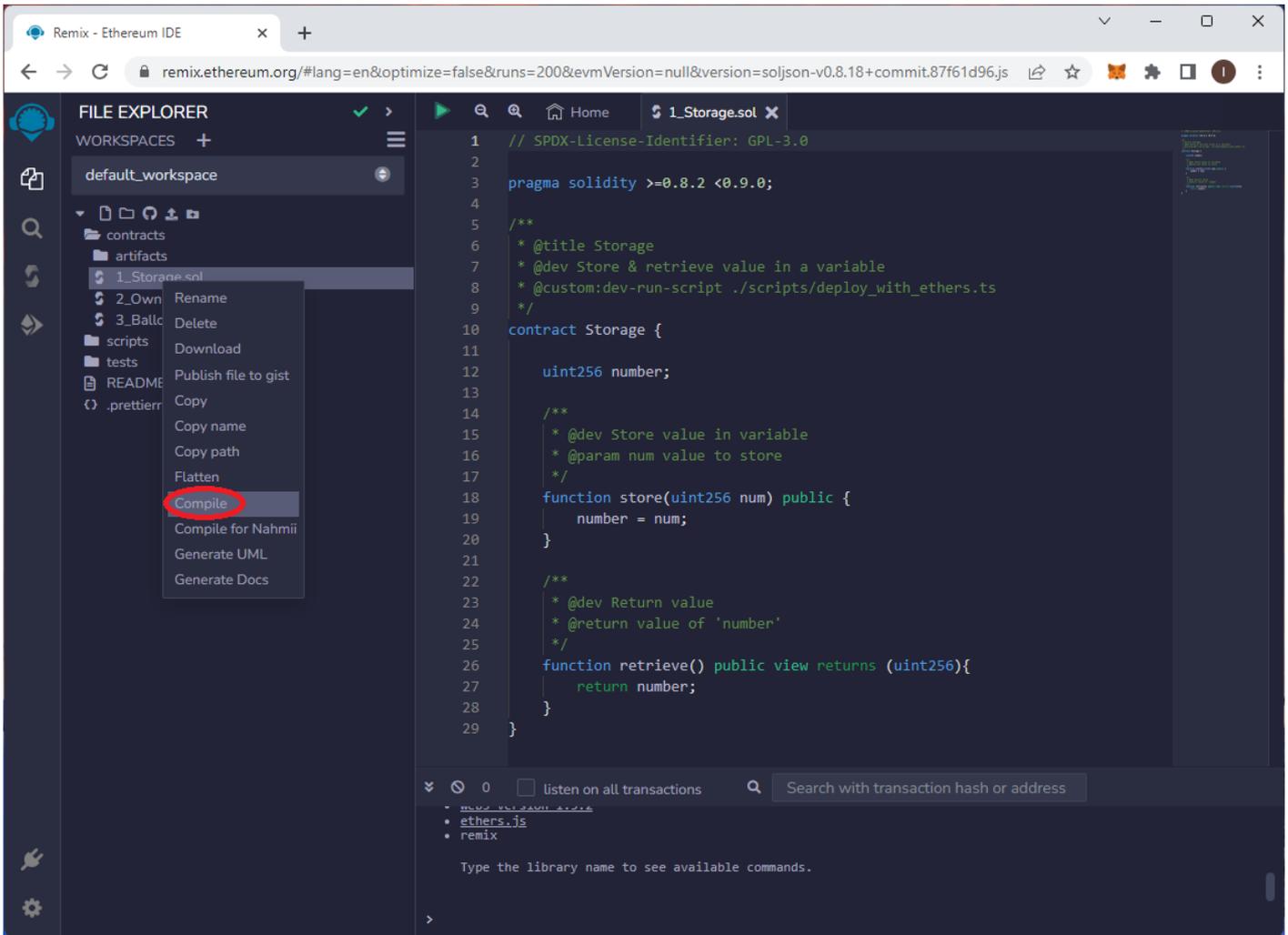
1. Open [Remix IDE](#).



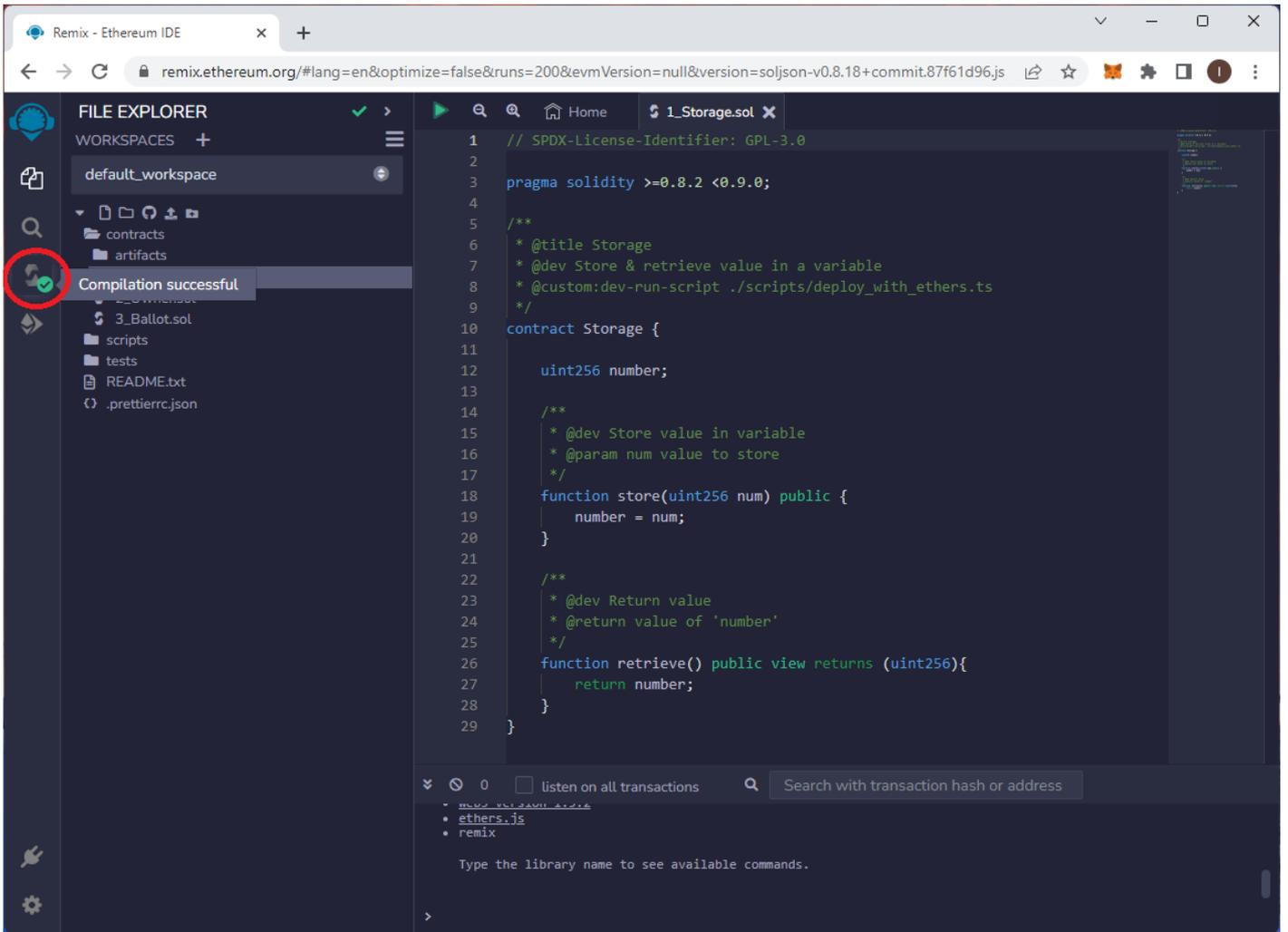
2. In our example, we will use one of the sample contracts from the IDE. This contract provides two functions: the first, `store`, allows you to save any number in the contract, while the second, `retrieve`, enables you to read the stored number. Open the `contracts/1_Storage.sol` file.



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



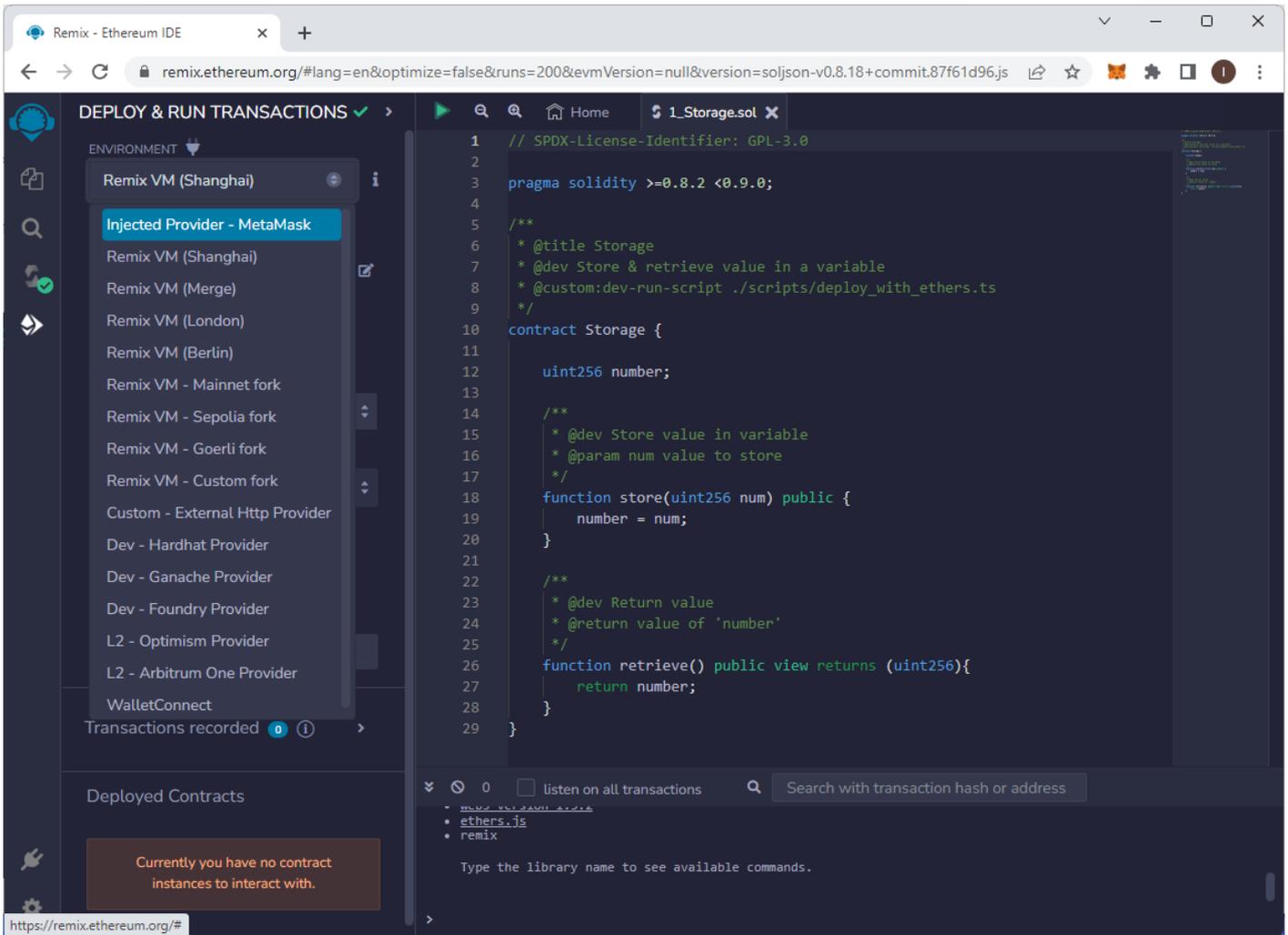
4. When the source code is error-free and the compilation is successful, you will see a green success icon.



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

The image shows the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is visible, showing the environment set to 'Remix VM (Shanghai)', the account as '0x5B3...eddC4 (100 ether)', and a gas limit of '3000000'. The value is set to '0' in 'Wei'. The contract selected is 'Storage - contracts/1_Storage.sol'. A red circle highlights the 'Deploy' button. The main editor shows the Solidity code for the 'Storage' contract, including a pragma statement, a license identifier, and two functions: 'store' and 'retrieve'. The bottom panel shows a file explorer with 'ethers.js' and 'remix' files, and a search bar for transaction hashes or addresses.

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

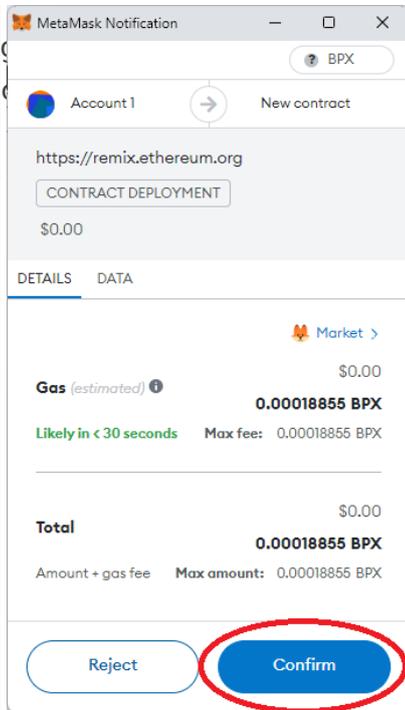
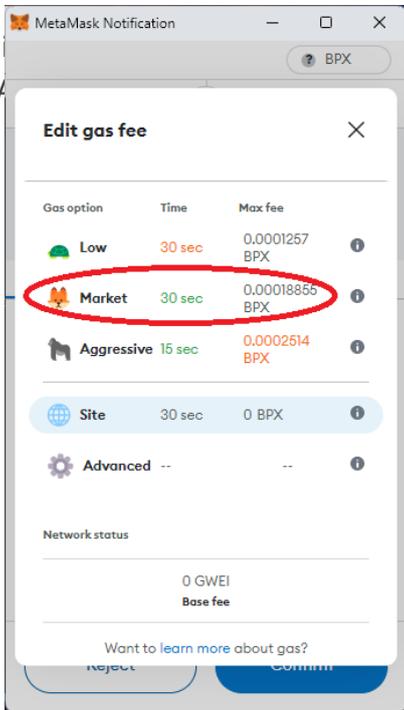
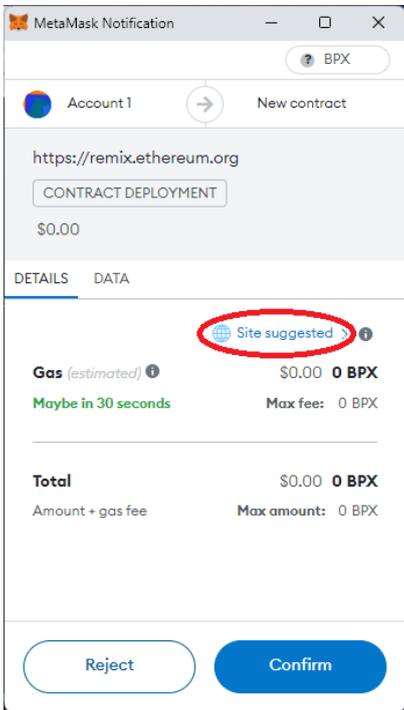
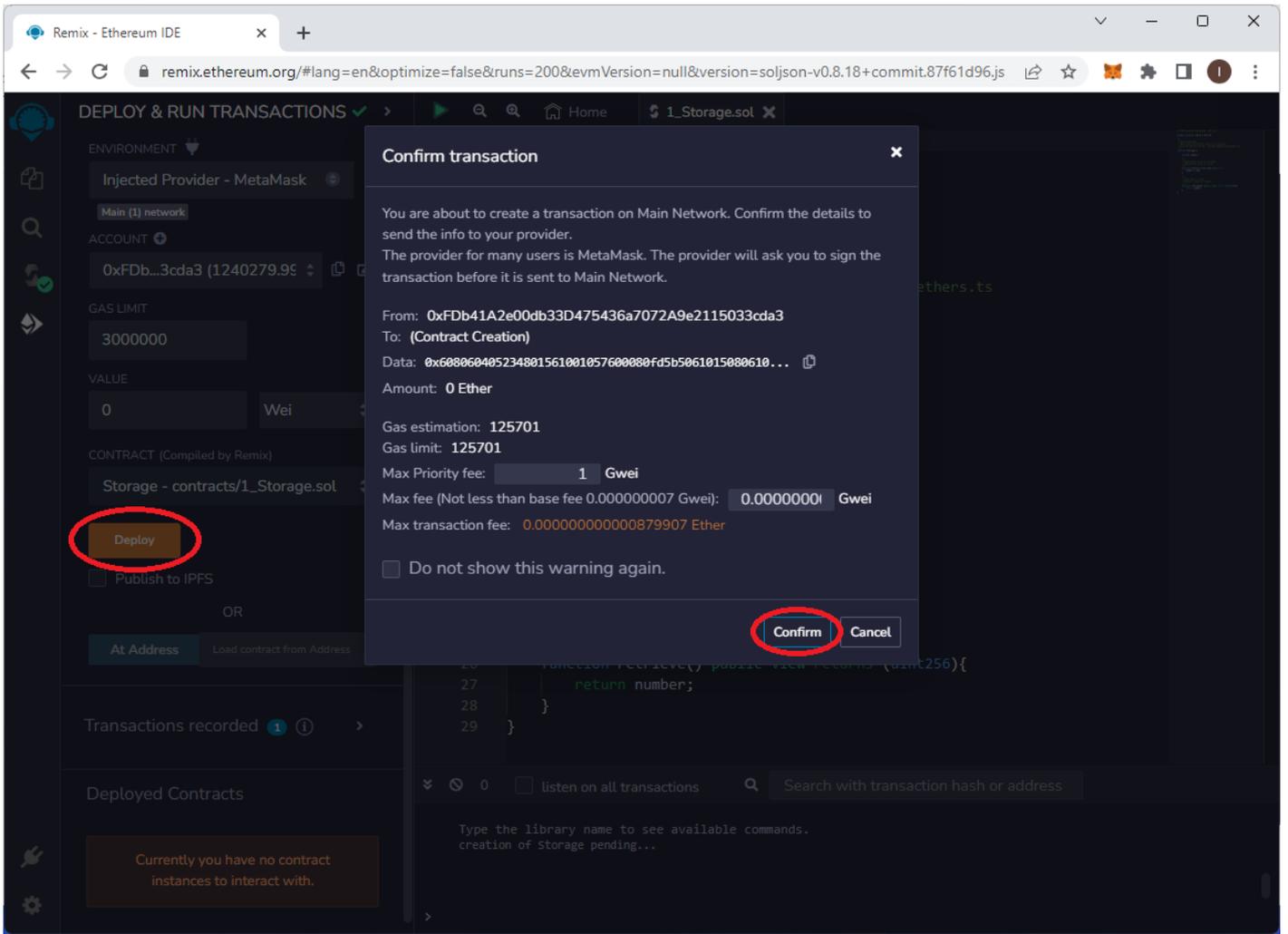
8. After successfully connecting, you should see your account address and its balance in the highlighted field.

The image shows the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is visible. Under 'ENVIRONMENT', 'Injected Provider - MetaMask' is selected. Under 'ACCOUNT', the account '0xFDb...3cda3 (1240279.95)' is highlighted with a red circle. The 'GAS LIMIT' is set to 3000000 and the 'VALUE' is 0 Wei. The contract 'Storage - contracts/1_Storage.sol' is selected. A 'Deploy' button is present. Below it, there are options for 'Publish to IPFS' and 'At Address'. The main editor shows the Solidity code for the 'Storage' contract:

```
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 }
```

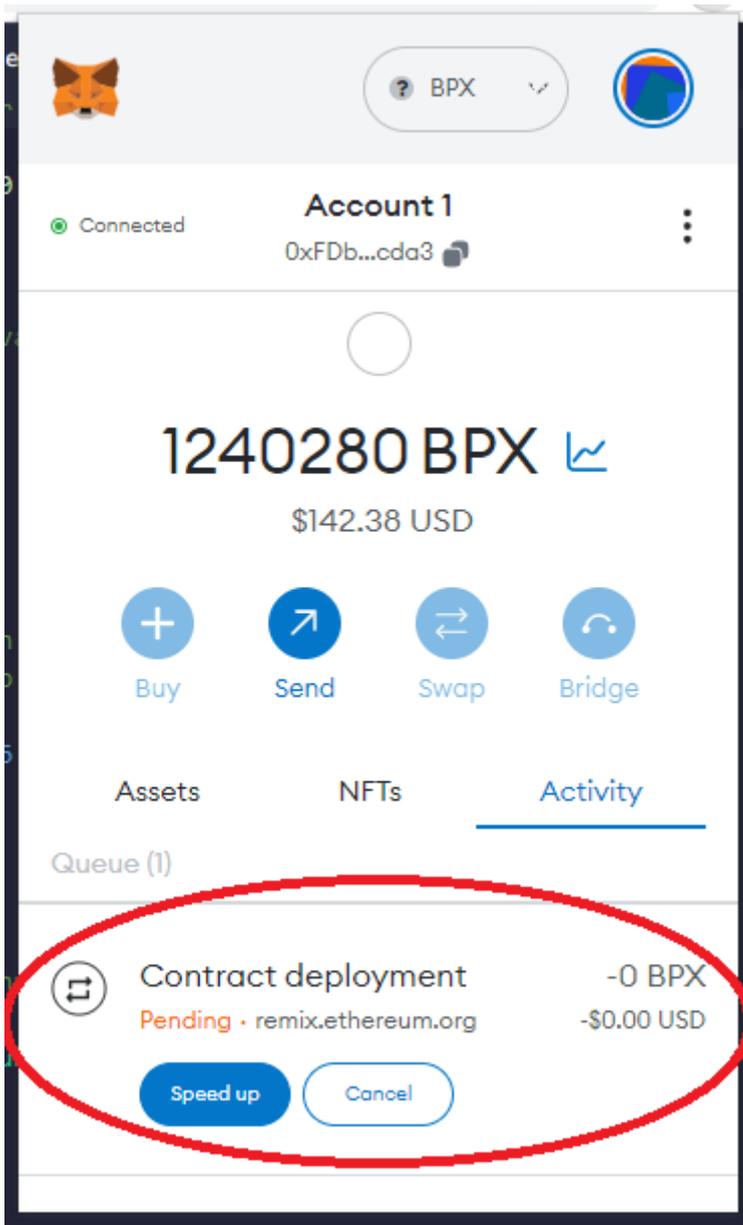
The bottom panel shows a file explorer with 'ethers.js' and 'remix' files. A search bar is present with the text 'Search with transaction hash or address'. Below it, there is a prompt: 'Type the library name to see available commands.'

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 image shows 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 is '0xFDb...3cda3 (1240279.99\$)' and the gas limit is '3000000'. The contract selected is 'Storage - contracts/1_Storage.sol'. The main editor displays the Solidity code for the 'Storage' contract, which includes a 'store' function and a 'retrieve' function. The bottom panel shows a transaction record for the 'Storage' contract at address '0x5CD...0E9B3', with a 'Debug' button highlighted in red.

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'. A 'Deploy' button is present, along with options to 'Publish to IPFS' or 'Load contract from Address'. Below this, there are sections for 'Transactions recorded' and 'Deployed Contracts'. The 'Deployed Contracts' section shows a contract at address 0x5CD...0E9B3, with a red circle around the 'view on etherscan' icon.

The main editor displays the Solidity code for the 'Storage' contract:

```
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 }
```

At the bottom, the transaction details are shown: '[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.

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

The image shows the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is visible, showing a value of '0' in Wei, the contract name 'Storage - contracts/1_Storage.sol', and a 'Deploy' button. Below this, there are options to 'Publish to IPFS' or 'Load contract from Address'. The 'Deployed Contracts' section shows a contract named 'STORAGE AT 0x5CD...0E9B3 (BLK)' with a balance of 0 ETH and a 'store' button next to a text field containing 'uint256 num'. The 'Low level interactions' section shows a 'Transact' 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 a transaction log with a green checkmark and the message: '[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 present in the bottom right of the transaction log.

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.

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 ✓

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 2 ⓘ

Deployed Contracts

STORAGE AT 0x5CD...0E9B3 (BLK)

Balance: 0 ETH

store 150

retrieve

Low level interactions

CALLDATA

Transact

```
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 }
```

0 listen on all transactions Search with transaction hash or address

view on etherscan

[block:27521 txIndex:0] from: 0xFDb...3cda3 to: Storage.store(uint256) 0x5CD...0e9B3 value: 0 wei data: 0x005...00096 logs: 0 hash: 0x8a5...01de0 Debug

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 image shows the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is visible, showing a 'Deploy' button and a 'Deployed Contracts' section. In the 'Deployed Contracts' section, a contract named 'STORAGE AT 0x5CD...0E9B3 (BLK)' is listed with a balance of 0 ETH. Below the contract name, there are two buttons: 'store' and 'retrieve'. The 'retrieve' button is circled in red. Below the buttons, the text '0: uint256: 150' is displayed. At the bottom of the panel, there is a 'Low level interactions' section with a 'Transact' button.

The main editor area shows the Solidity code for the 'Storage' contract:

```
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 }
```

At the bottom of the IDE, a debug console shows a call: '[call] from: 0xFDb41A2e00db330475436a7072A9e2115033cda3 to: Storage.retrieve() data: 0x2e6...4cec1'. A 'Debug' button is visible next to the call.

Revision #3

Created 8 June 2023 07:45:26 by Admin

Updated 5 November 2024 12:05:12 by Admin