Introduction to Solidity

Deployment in Remix

Copy page

Use Remix to deploy and interact with a contract.

Remix contains a simulation of the blockchain that allows you to easily and safely deploy and interact with contracts, for free.

Objectives

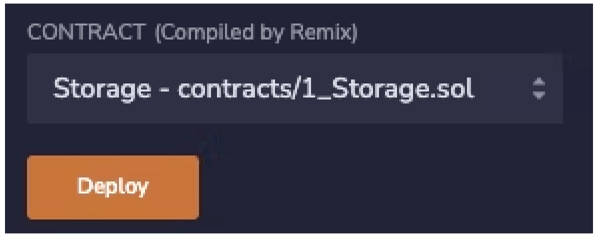
By the end of this lesson you should be able to:

- Deploy and test the Storage.sol demo contract in Remix

Deploy the `Storage` Contract

Deploying a contract is easy, but remember that if the contract doesn't compile, the deploy button will instead deploy the last good compiled version of your contract. Verify that you see a green checkmark on the icon for the *Compiler* contract on the left side of the editor, then select the *Deploy & Run Transactions* plugin.

If you've already deployed any contracts, press the trash can button to the right of the *Deployed Contracts* label. Then, press the orange button to deploy the `Storage` contract.


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

Contract Addresses

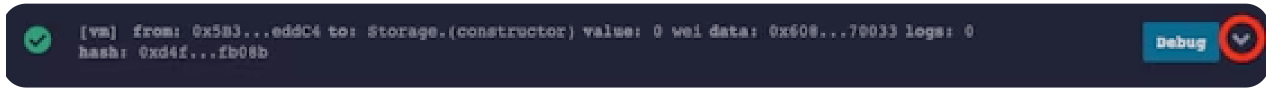
After your contract deploys, it will appear in the *Deployed* contracts section as *STORAGE AT* followed by an address. Addresses are used for both contracts and wallets in EVM-compatible blockchains and serve a similar purpose to an IP address. You can copy the address to see what it looks like. It's 20 characters of hexadecimal, similar to `0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8` .

The address is what you will use to find your contract with tools such as *Etherscan*, or to connect to it with a front end.

However, when you deploy using the Remix VM simulation, it will only exist in your browser.

Deployments and Test Accounts

The result of any transactions, including deployments, will appear in the Remix terminal. Click the chevron next to the blue *Debug* button to expand the log.



```
[vm] from: 0x5B3...eddC4 to: Storage.(constructor) value: 0 wei data: 0x608...70033 logs: 0
hash: 0xd4f...fb08b
```

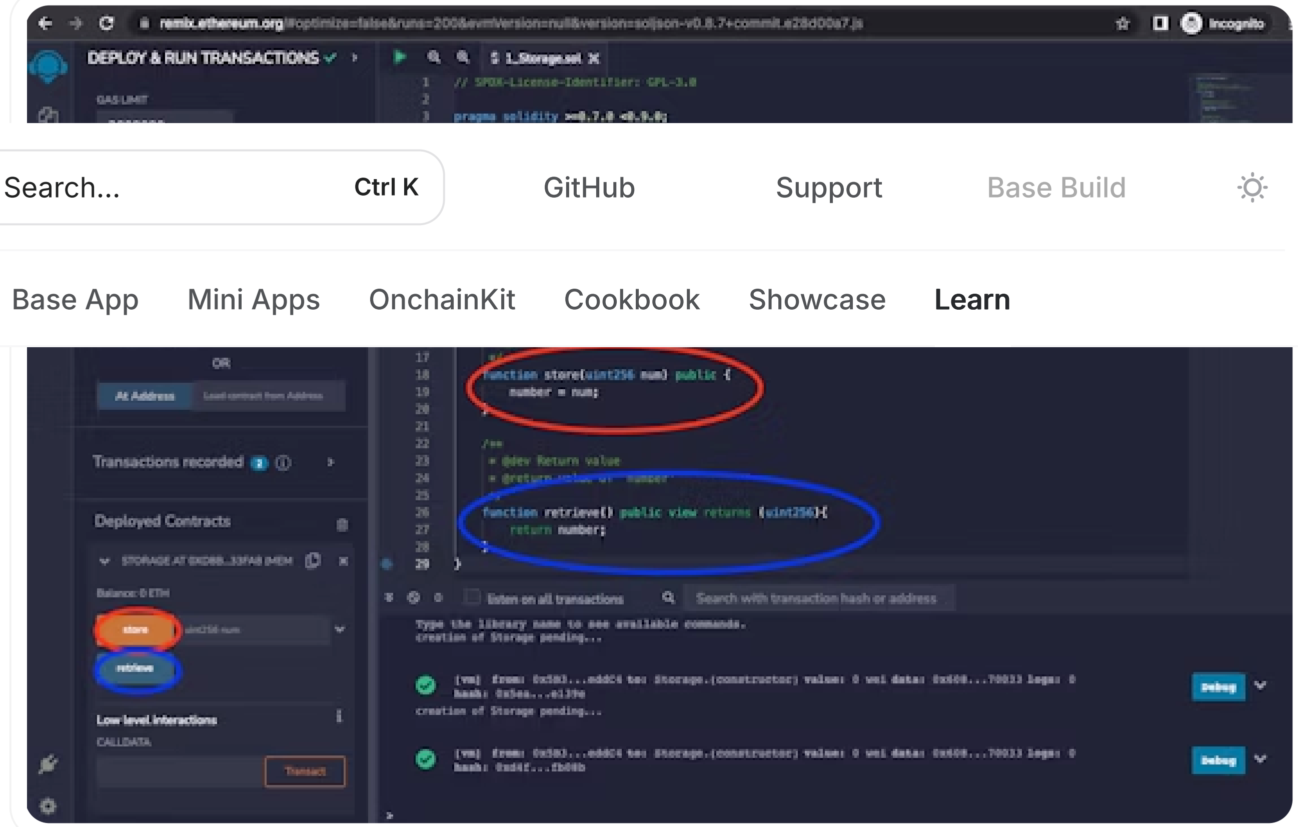
Doing so will show the full transaction log, which contains all of the details of the transaction, such as its amount, the address to and from, and the inputs and outputs provided to the transaction.

In this case, the sender (from) matches the first listed account in the panel, which has spent a small amount of simulated Ether to deploy the contract.

You can access a list of 15 test wallets here, each with 100 Ether to spend. Among other uses, you can use these accounts to compare behavior between wallets that are and are not the owner of a deployed contract.

Interacting with the Contract

Click the chevron to expand your contract in the Deployed Contracts section of the left panel. You'll see two buttons, one for each `public` function in the `Storage` contract. Notice how the *Store* button also has a field to pass a *uint256*, matching the parameter for `uint256 num`.



```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0;
...
    function store(uint256 num) public {
        number = num;
    }

    /**
     * @dev Return value
     * @return value of 'number'
     */
    function retrieve() public view returns (uint256){
        return number;
    }
}
```

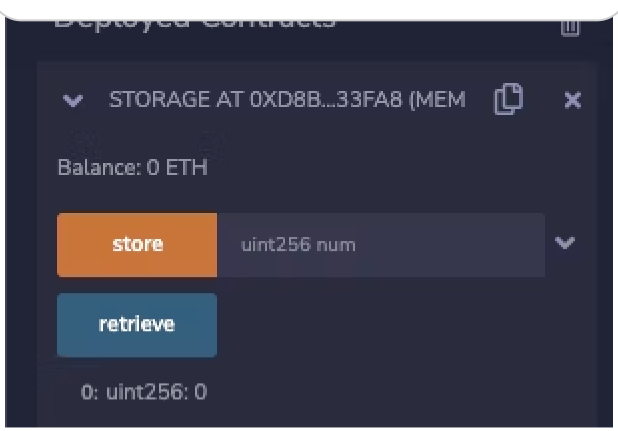
```
Type the library name to see available commands.
creation of Storage pending...

[vm] from: 0x5B3...eddC4 to: Storage.(constructor) value: 0 wei data: 0x608...70033 logs: 0
hash: 0x5ea...e139e
creation of Storage pending...

[vm] from: 0x5B3...eddC4 to: Storage.(constructor) value: 0 wei data: 0x608...70033 logs: 0
hash: 0xd4f...fb08b
```

Let’s click the retrieve button first. Before clicking, make a prediction: given that the `number` variable was instantiated without a value, what do you think the return will be?

Go ahead and click – the result will appear below the button as:

```
0: uint256: 0
```

Ask a question... Ctrl+I

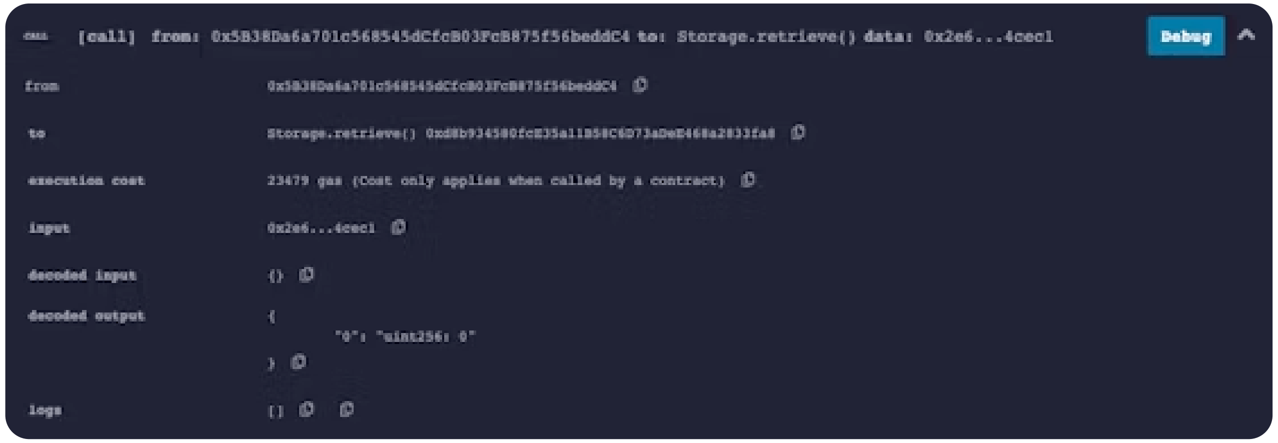


0: uint256: 0

Outputs from the EVM are in the form of an array, so in this case, the only return is in the 0th element and it is a `uint256` of 0. Were you expecting `undefined` or an error?

Unlike many languages, variables in Solidity have a **default value** if not assigned. For `uint` and `int` , that value is 0.

You can also review the results of your transaction in the console.



```
CALL  [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Storage.retrieve() data: 0x2e6...4cec1
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  Storage.retrieve() 0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8
execution cost      23479 gas (Cost only applies when called by a contract)
input               0x2e6...4cec1
decoded input       {}
decoded output      {
                        "0": "uint256: 0"
                    }
logs                []
```

The screenshot above is from a newer version of Remix than the video. Outputs are now often decoded for you!

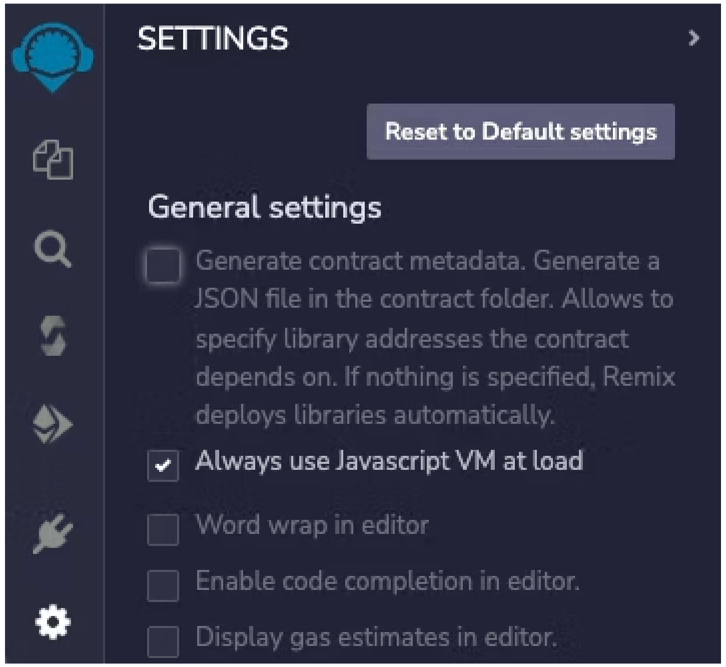
Storing and Retrieving a Value

Use the input to store and retrieve a value. Which costs more gas? Storing or retrieving? This isn't a trick question, but it is a bit nuanced. Both cost about 23500 gas, but there is only a gas cost for the retrieve function if it is called by another contract. Calling it from the web is free, because you're only reading data that is on the blockchain and are not asking the EVM to perform a computational task.

Disabling Artifact Generation

Return to the *File Explorer* by clicking the double document icon in the upper left. You should now see a folder called *artifacts* that has been added to your project. This folder contains a number of build artifacts, such as the ***ABI*** for your contract, that will be useful to you later, but currently just cause clutter.

You can disable artifact generation by clicking the settings gear in the bottom left corner, then deselecting the first checkbox to *Generate contract metadata...*



- [ ] Generate contract metadata. Generate a JSON file in the contract folder. Allows to specify library addresses the contract depends on. If nothing is specified, Remix deploys libraries automatically.
- [✓] Always use Javascript VM at load
- [ ] Word wrap in editor
- [ ] Enable code completion in editor.
- [ ] Display gas estimates in editor.

Conclusion

Remix makes it easy to write, deploy, and test contracts. Contracts are deployed by a wallet address to their own address. These addresses are similar to how IP addresses work, in that they enable connections across the network. You can test deployed contracts directly in Remix and use the console to see detailed information about each transaction.

Was this page helpful? Yes No Suggest edits Raise issue

‹ Deployment in Remix Introduction to Contracts ›