Counter Smart Contract

Remix is a powerful, open source tool that helps you write Solidity contracts straight from the browser. Remix also supports testing, debugging and deploying of smart contracts and much more.

It is recommended to useRemix to create and deploy smart contracts quickly.

In Remix, when opening a basic template you will see a folder called contracts. Right click on the folder and create a new contract called Counter.sol and begin typing the code below.

License

At the start of your smart contract, you can declare a license. In this case, we will leave this as UNLICENSED as we are not too concerned about what this code is used for.

// SPDX-License-Identifier: UNLICENSED

Mention Compiler Version

pragma solidity ^0.8.9; Make sure the solidity version declared in the contract matches your compiler version Now let's write out our contract:

contract

Counter

{ uint256

public count ;

event

updateCount (uint newCount) ;

function

incrementCount ( )

public

returns (uint256)

{ count += 1 ; emit

updateCount (count) ; return count ; } } Here is a breakdown of what this contract is doing:

1. uint256 public count
2. : This line declares a public unsigned integer variable
3. named count. Unsigned integers are non-negative numbers. Thepublic
4. keyword
5. automatically generates a function that allows you to access the current
6. value of thecount
7. state variable from outside the contract.
8. event updateCount(uint newCount)
9. : This line declares an event namedupdateCount
10. . Events are a way for your contract to communicate that
11. something happened on the blockchain to your app front-end, which can be
12. 'listening' for certain events and take action when they happen. In this
13. case, the event will emit the new value ofcount
14. .
15. function incrementCount() public returns(uint256)
16. : This line declares a
17. public function named incrementCount that doesn't take any parameters and
18. returns an unsigned integer. Inside the incrementCount function:
19. count +=1
20. : This line increases thecount

21. variable by one.
22. emit updateCount(count)
23. : This line triggers theupdateCount
24. event and
25. sends the new value of the count variable to the listener.
26. return count
27. : This line returns the new value ofcount
28. .

Now that we have our basic contract let's deploy this to the Polygon Mumbai Test Network. You can follow the officialRemix Documentation for Deployment to get this deployed and verified.Previous Introduction Next Using Paymaster and Biconomy Dashboard