How to test and deploy smart contacts on Remix IDE?

1. Go to the Remix online IDE.

https://remix.ethereum.org/

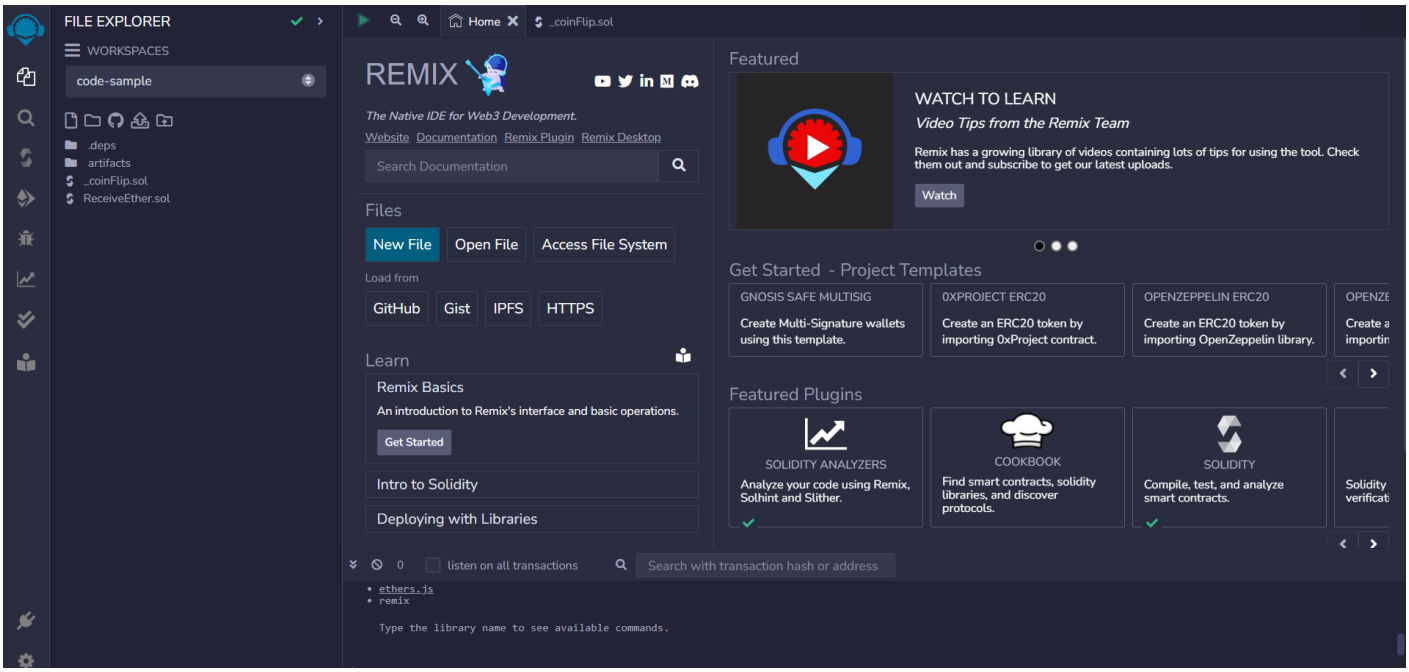

Fig. 1 Remix IDE

2. Write a Smart Contract

Create a Solidity file.The extension used for solidity language in which our smart contract is being written is '.sol'

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.7.0 <0.9.0;

contract CoinFlip {
    address public sender;
    address public player1;
    address public player2;
    uint256 public balance;

    constructor(address _player1, address _player2) {
        sender = msg.sender;
        player1 = _player1;
        player2 = _player2;
        balance = 0;
    }

    function addEther() external payable {
        require(msg.sender == sender, "Only the sender can add ether.");
        balance += msg.value;
    }

    function distribute() external {
        require(msg.sender == sender, "Only the sender can distribute.");

        uint256 lastDigit = block.timestamp % 10;

        if (lastDigit % 2 == 0) {
            payable(player1).transfer(balance);
        } else {
            payable(player2).transfer(balance);
```

```
        }

        balance = 100;
    }
}
```

3. Compile the contract

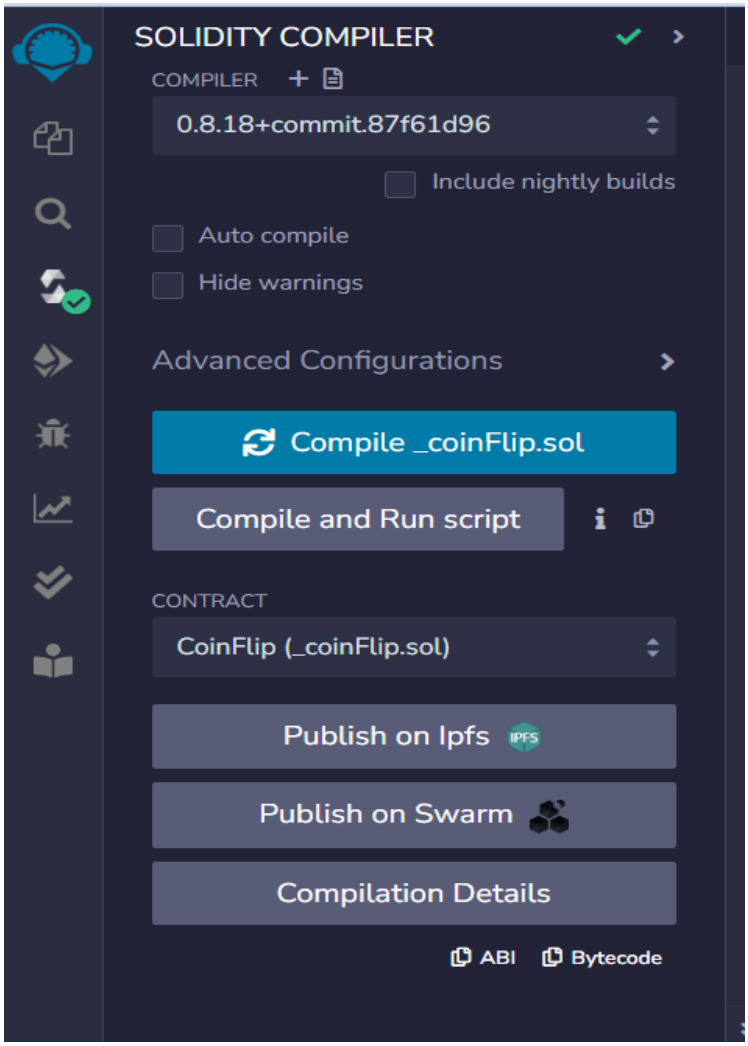

Fig. 2 Compiling the contract

4. Deploy the contract

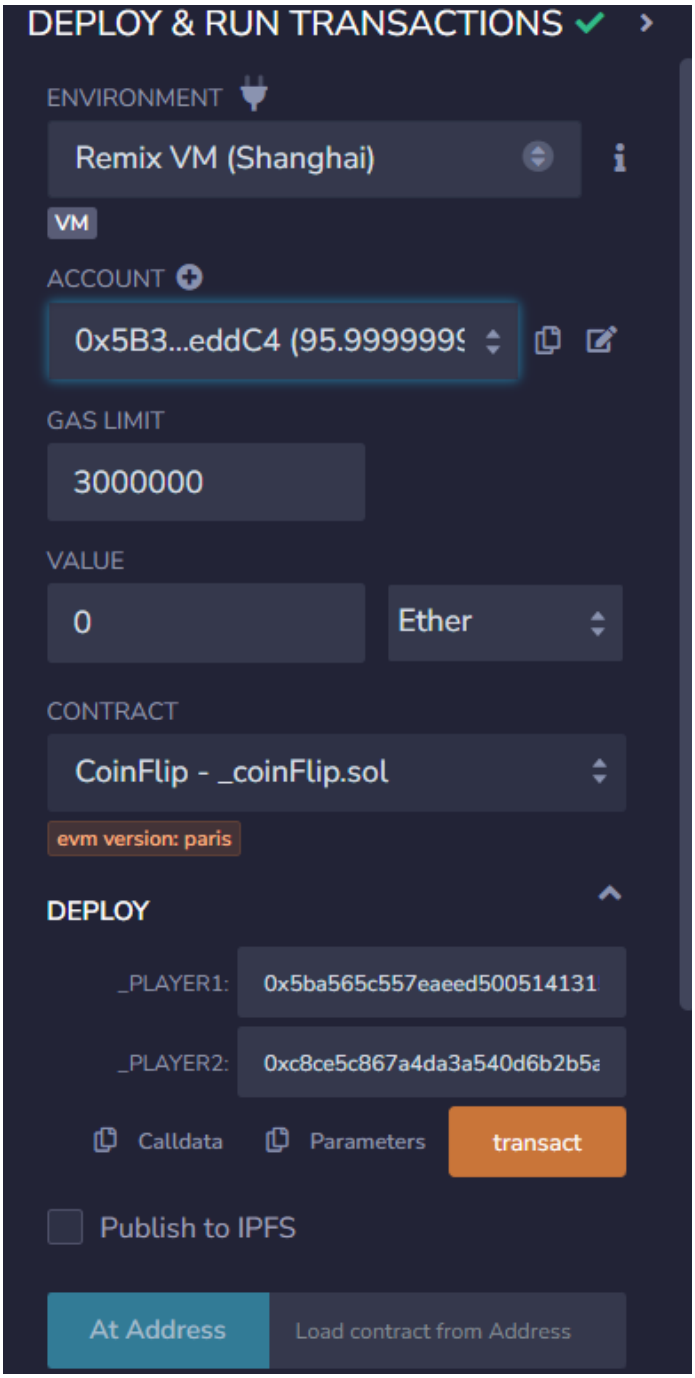

Fig. 3 Deploying the contract

Head to the Deploy and run transactions section.
For our coin flip contract, we have 2 players and a sender. When deploying the contract, we will need to select an address for the Sender, and the addresses for players 1 and 2. The provided accounts are just test ids and are only meant for testing purposes.
Click on transact to deploy the contract in the test environment!

5. **Test the contract.**

To test this contract we will need to send test Ether from the Sender's account to the balance of the smart contract. To do this, I have selected 4 Ethers and clicked 'addEther' function to add this balance (see fig 4).

When I click on the ‘balance’, I see 4 ETH as the current balance. Next, the ‘distributeEther’ function will randomly select either a 0 or a 1 and distribute the balance to either player 1 or 2, based on which play wins. The last figure shows how the sender’s balance has been reduced and 4 ETH has been added to Player 1’s account.

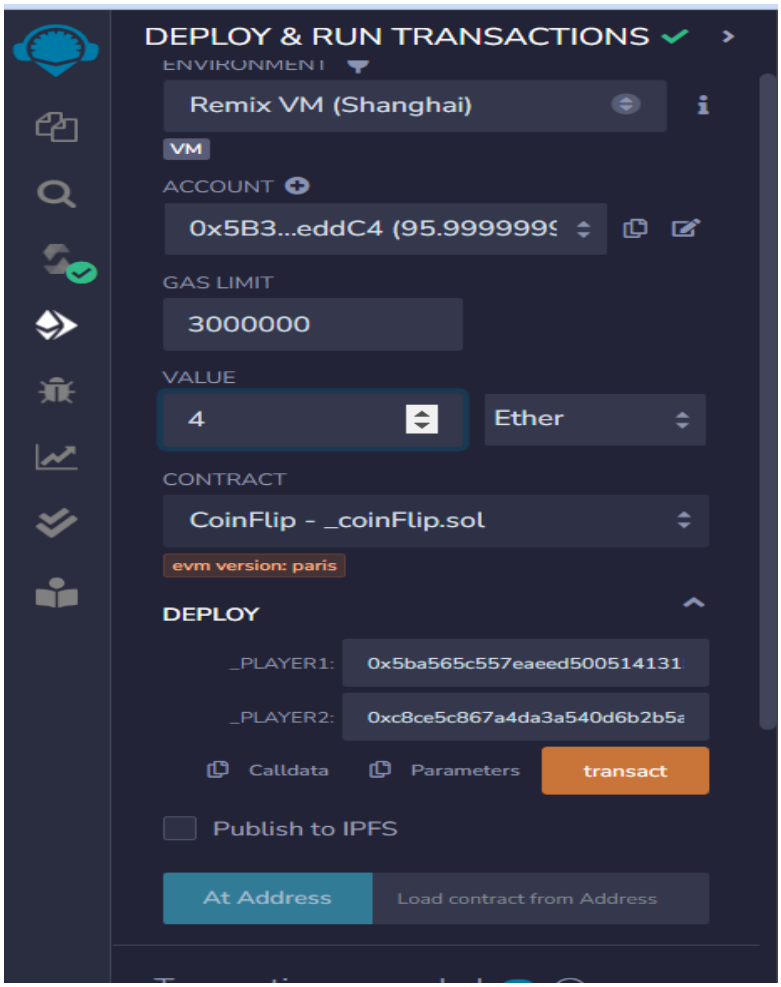

Fig. 4 Adding ETH to the contract

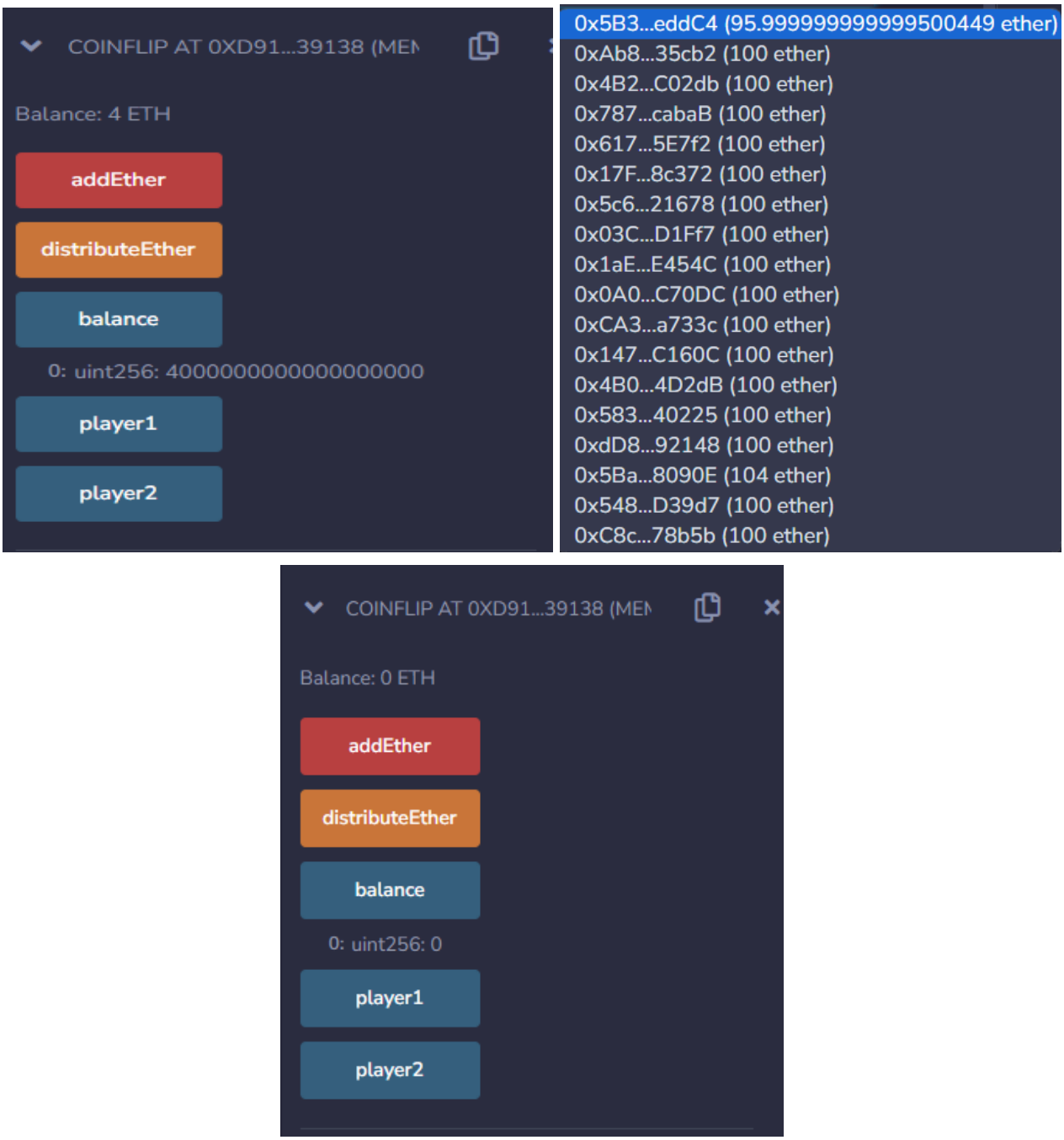

Fig. 7 Contract balance after distribution

By following these steps, we have successfully tested a smart contract.