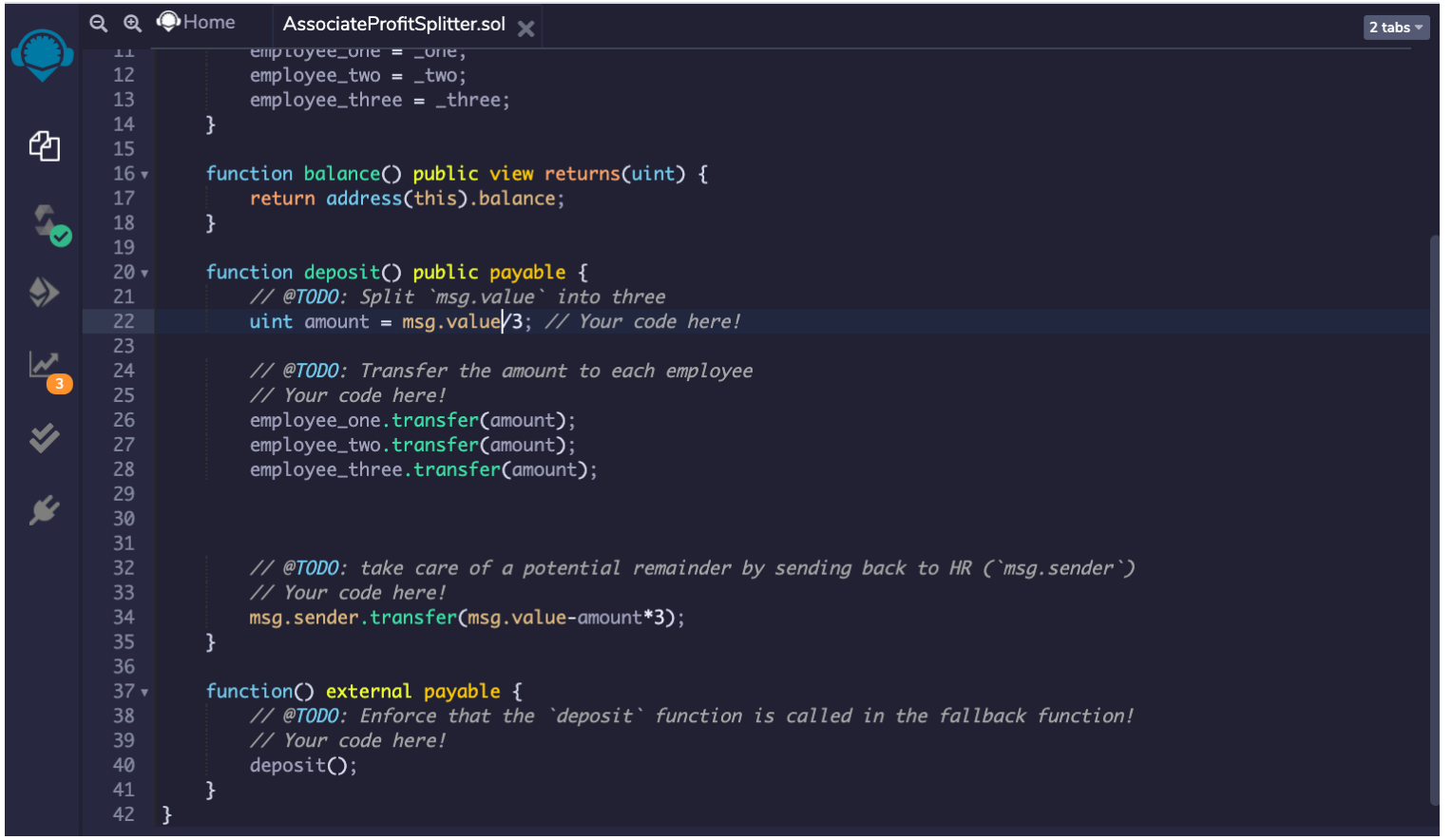
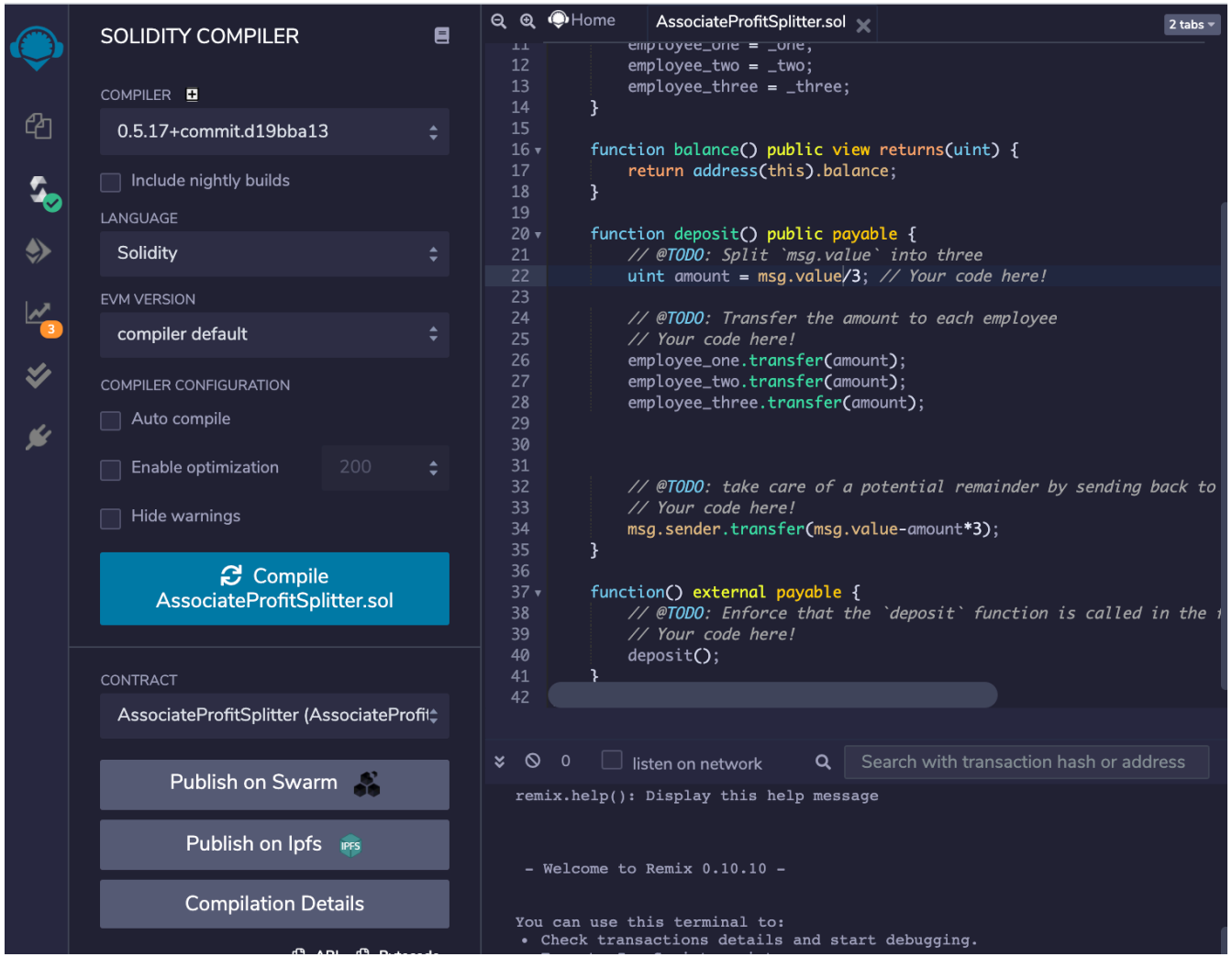
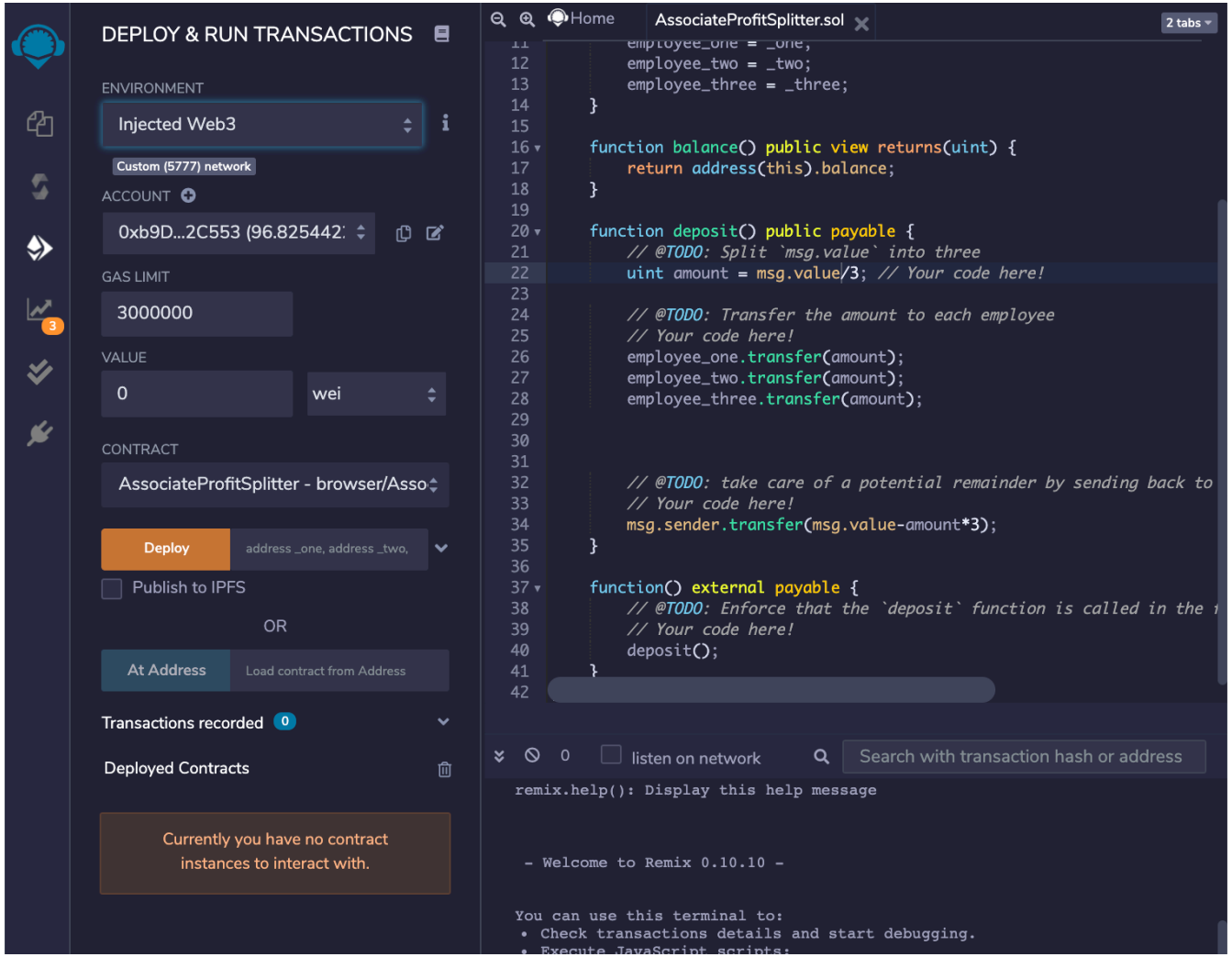
Create the first contract



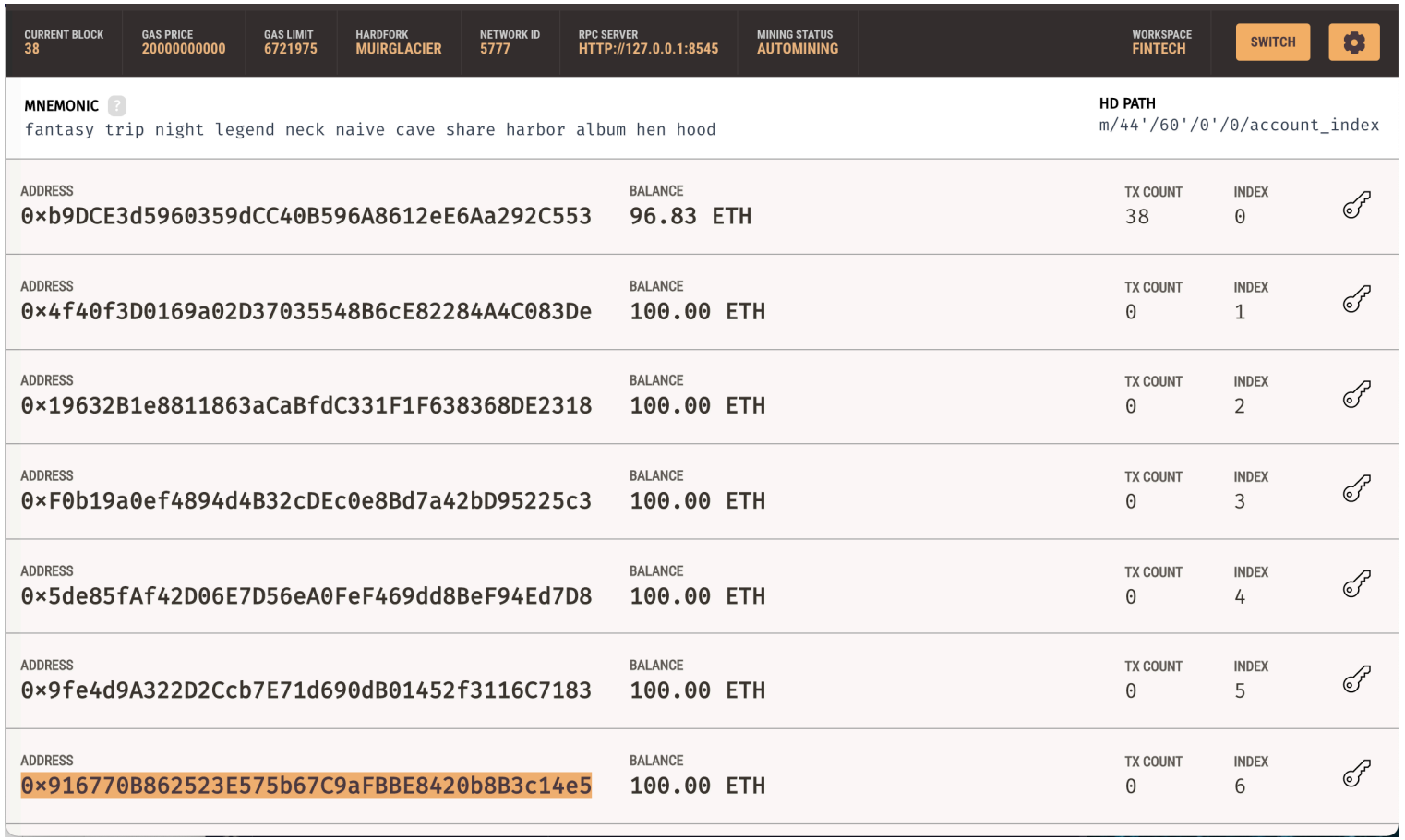
Compiling first contract



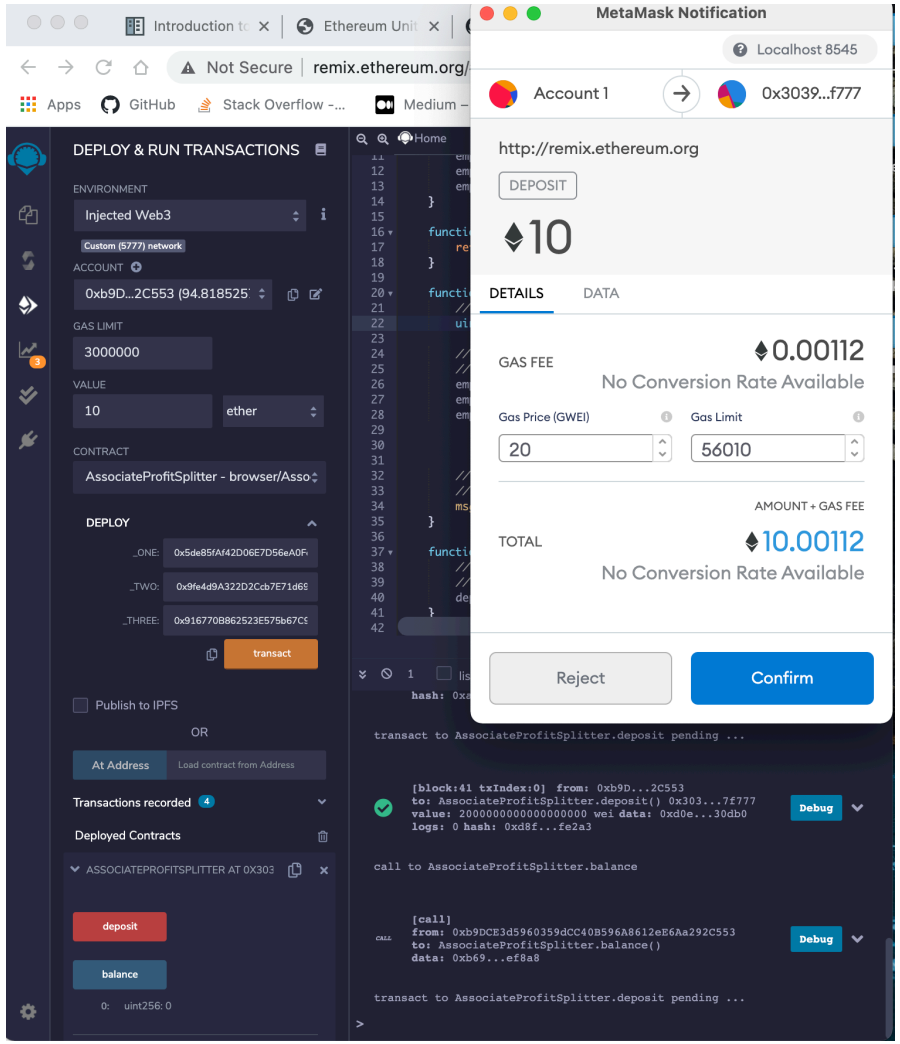
Deploy the first contract



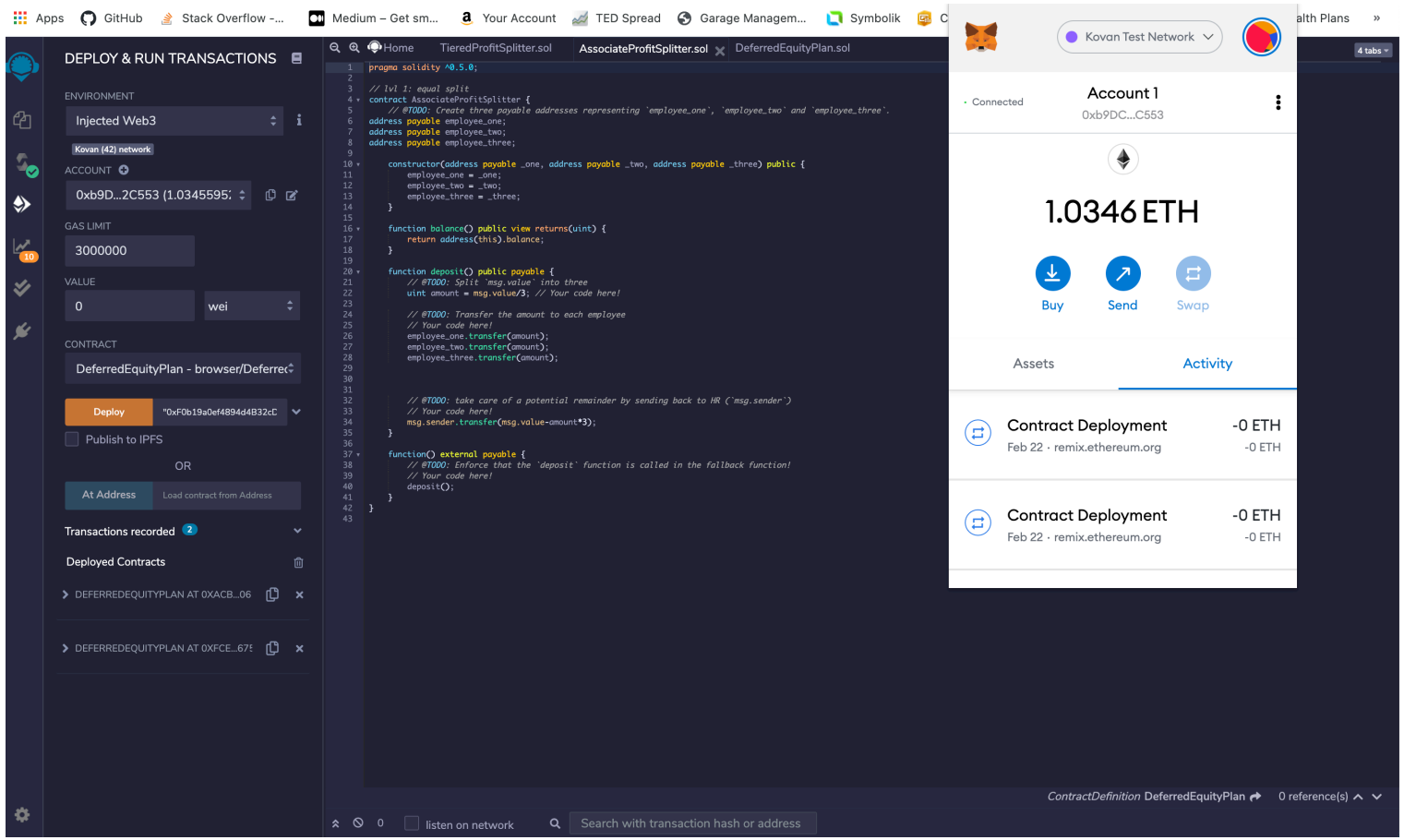
Addresses used for the Employees



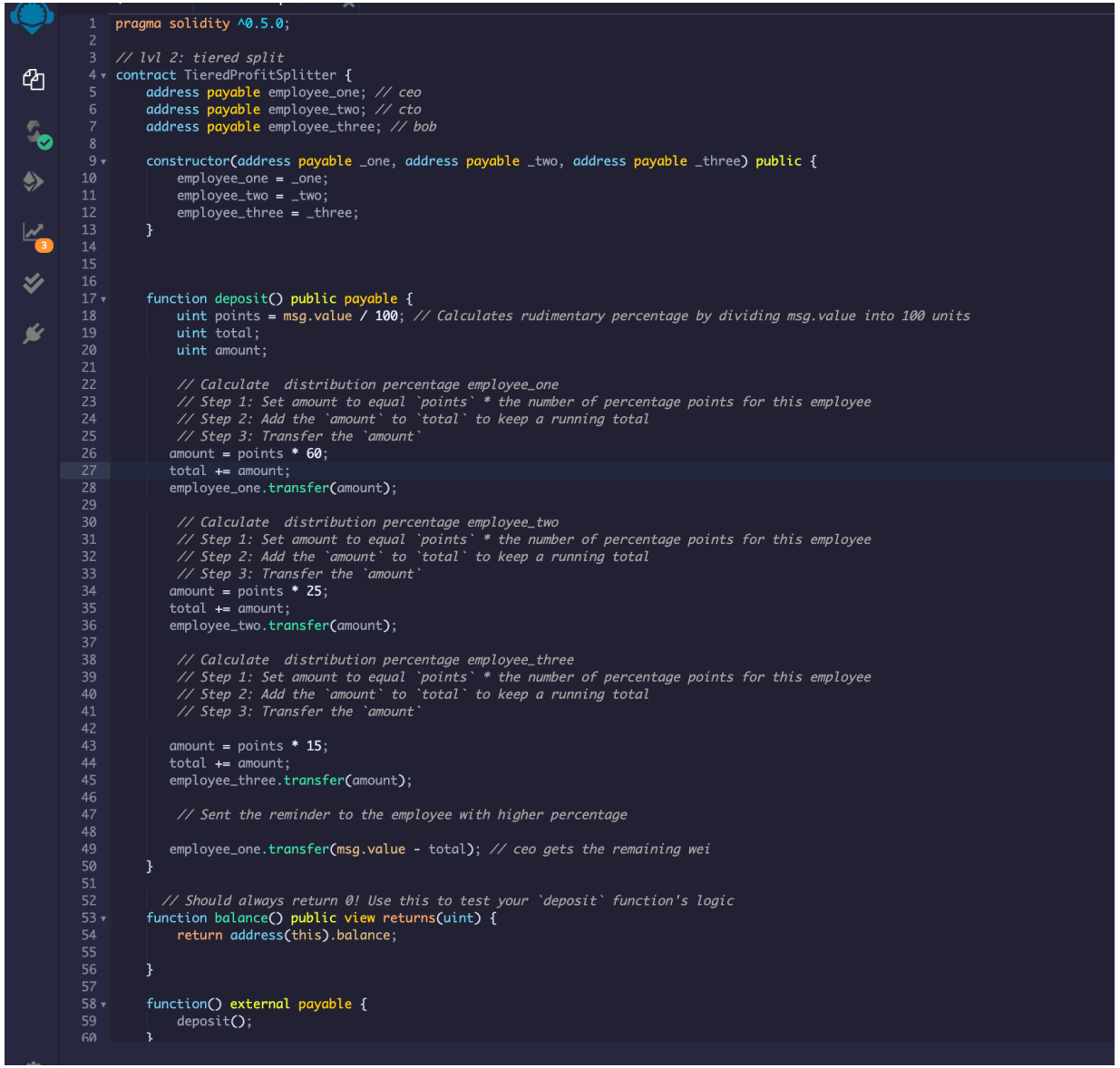
First transaction



Deploy the first contract in live Tesnet

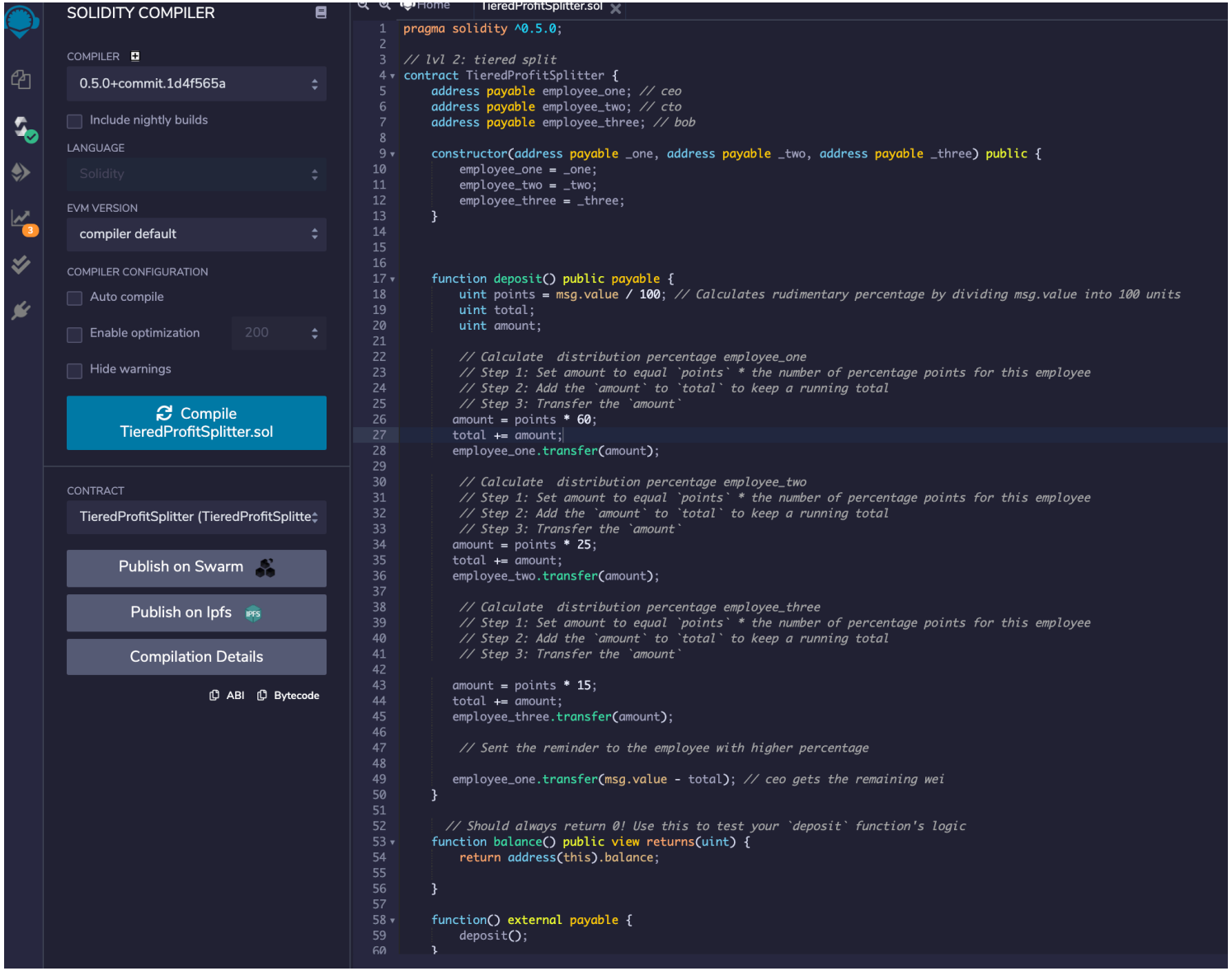


Create second contract

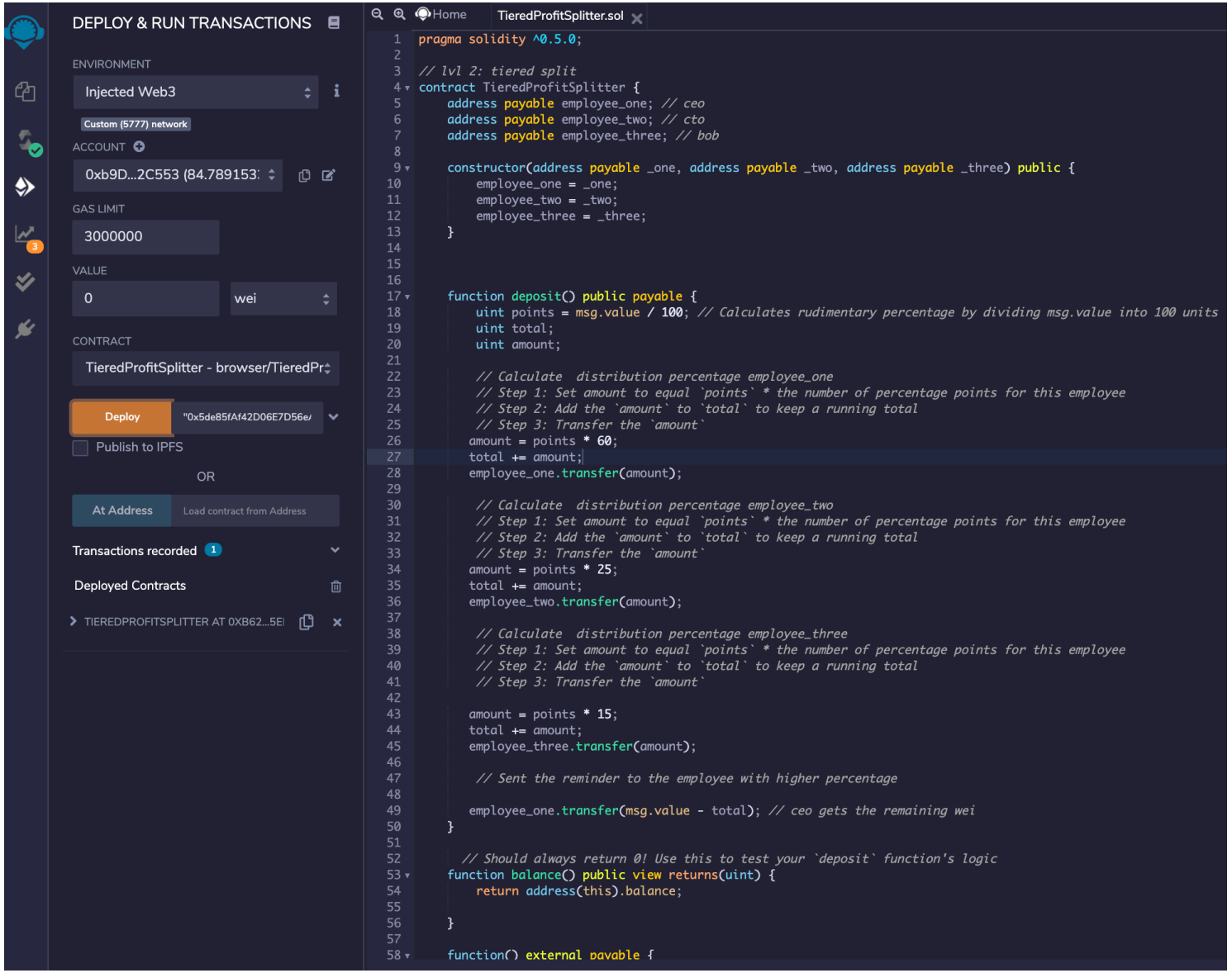


```
pragma solidity ^0.5.0;

// lvl 2: tiered split
contract TieredProfitSplitter {
    address payable employee_one; // ceo
    address payable employee_two; // cto
    address payable employee_three; // bob

    constructor(address payable _one, address payable _two, address payable _three) public {
        employee_one = _one;
        employee_two = _two;
        employee_three = _three;
    }



    function deposit() public payable {
        uint points = msg.value / 100; // Calculates rudimentary percentage by dividing msg.value into 100 units
        uint total;
        uint amount;

        // Calculate  distribution percentage employee_one
        // Step 1: Set amount to equal `points` * the number of percentage points for this employee
        // Step 2: Add the `amount` to `total` to keep a running total
        // Step 3: Transfer the `amount`
       amount = points * 60;
       total += amount;
       employee_one.transfer(amount);

        // Calculate  distribution percentage employee_two
        // Step 1: Set amount to equal `points` * the number of percentage points for this employee
        // Step 2: Add the `amount` to `total` to keep a running total
        // Step 3: Transfer the `amount`
       amount = points * 25;
       total += amount;
       employee_two.transfer(amount);

        // Calculate  distribution percentage employee_three
        // Step 1: Set amount to equal `points` * the number of percentage points for this employee
        // Step 2: Add the `amount` to `total` to keep a running total
        // Step 3: Transfer the `amount`

       amount = points * 15;
       total += amount;
       employee_three.transfer(amount);

        // Sent the reminder to the employee with higher percentage

       employee_one.transfer(msg.value - total); // ceo gets the remaining wei
    }

      // Should always return 0! Use this to test your `deposit` function's logic
    function balance() public view returns(uint) {
        return address(this).balance;

    }

    function() external payable {
        deposit();
    }
```

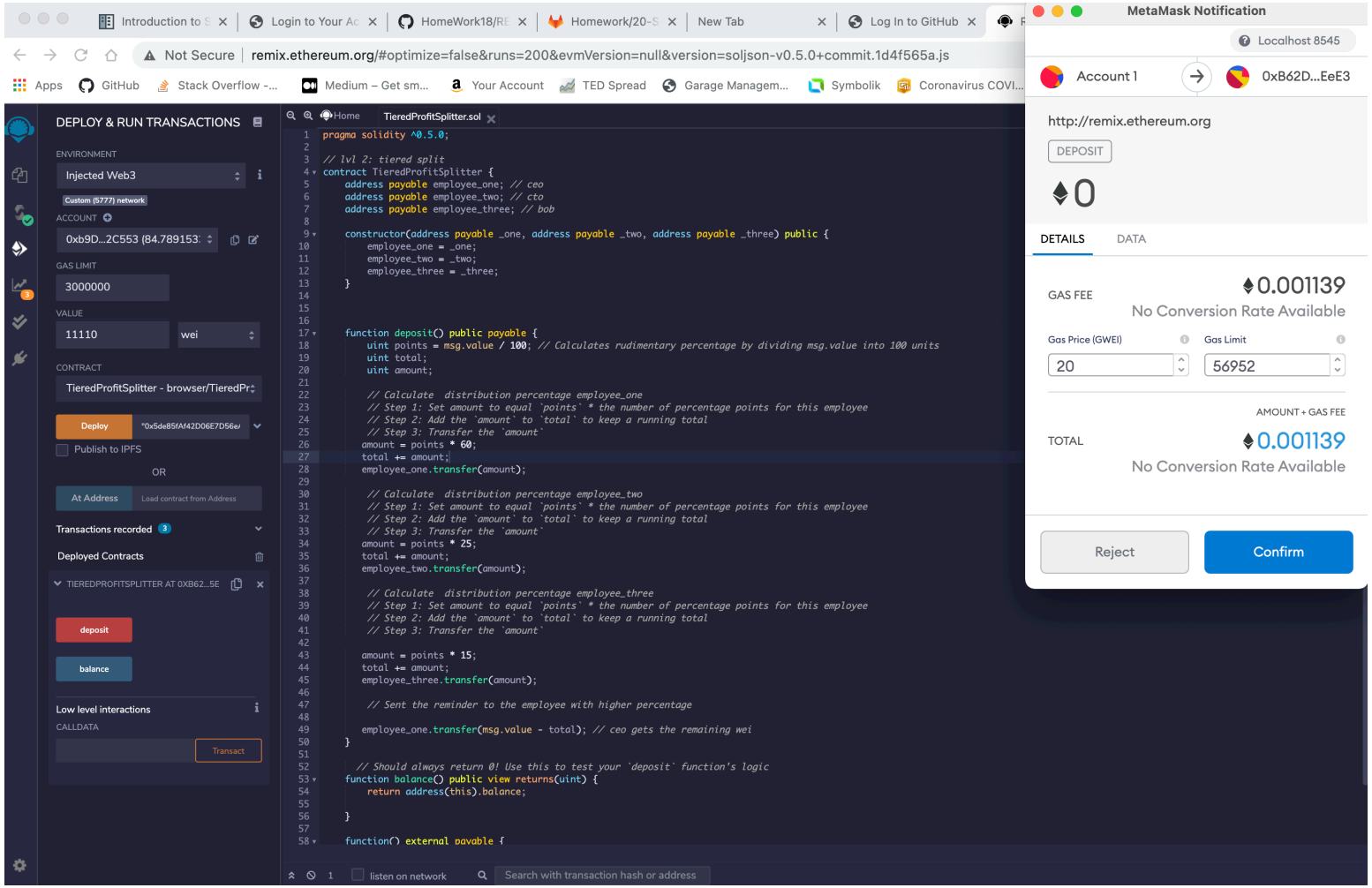
Compile second contract



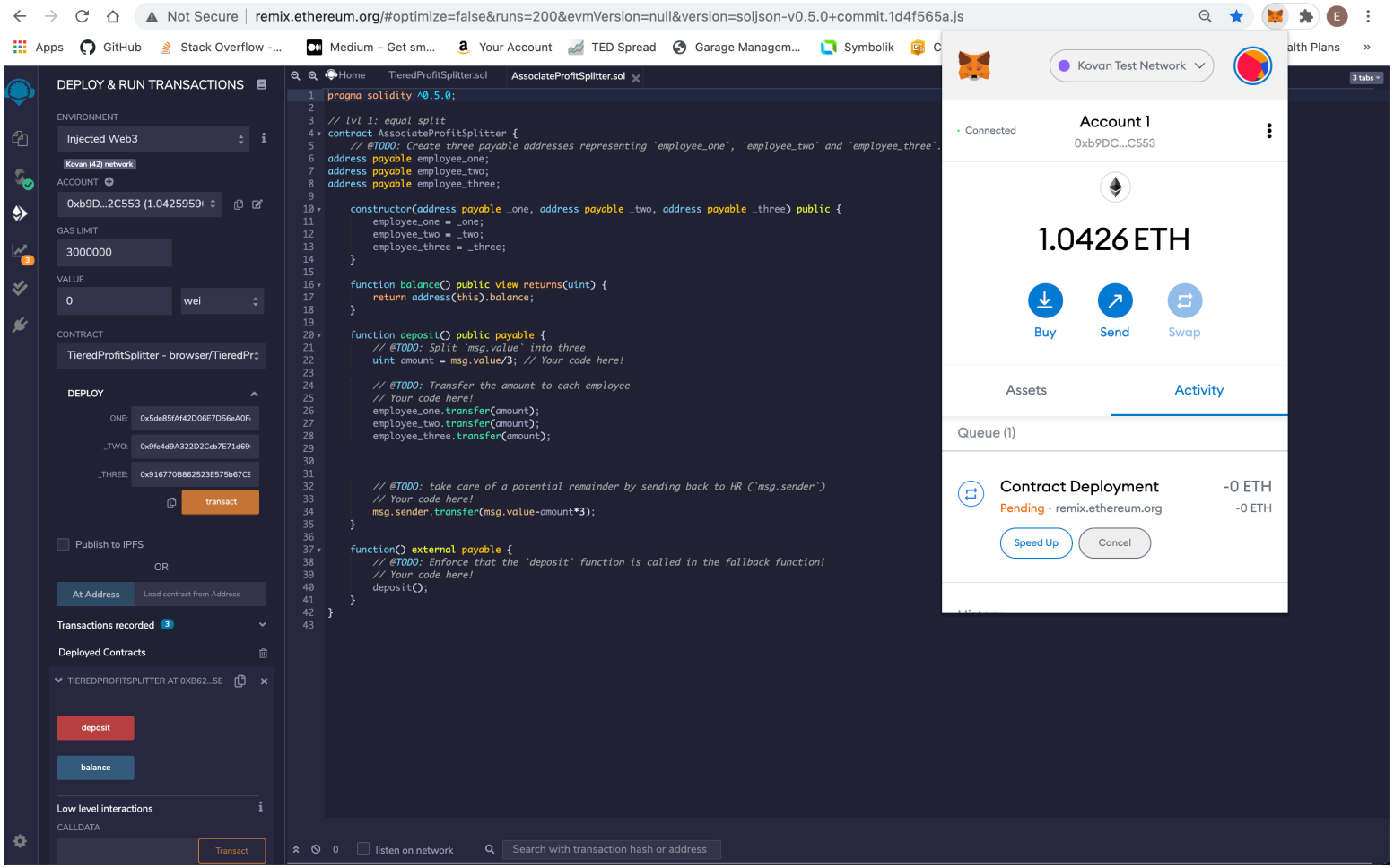
Deploy Second contract



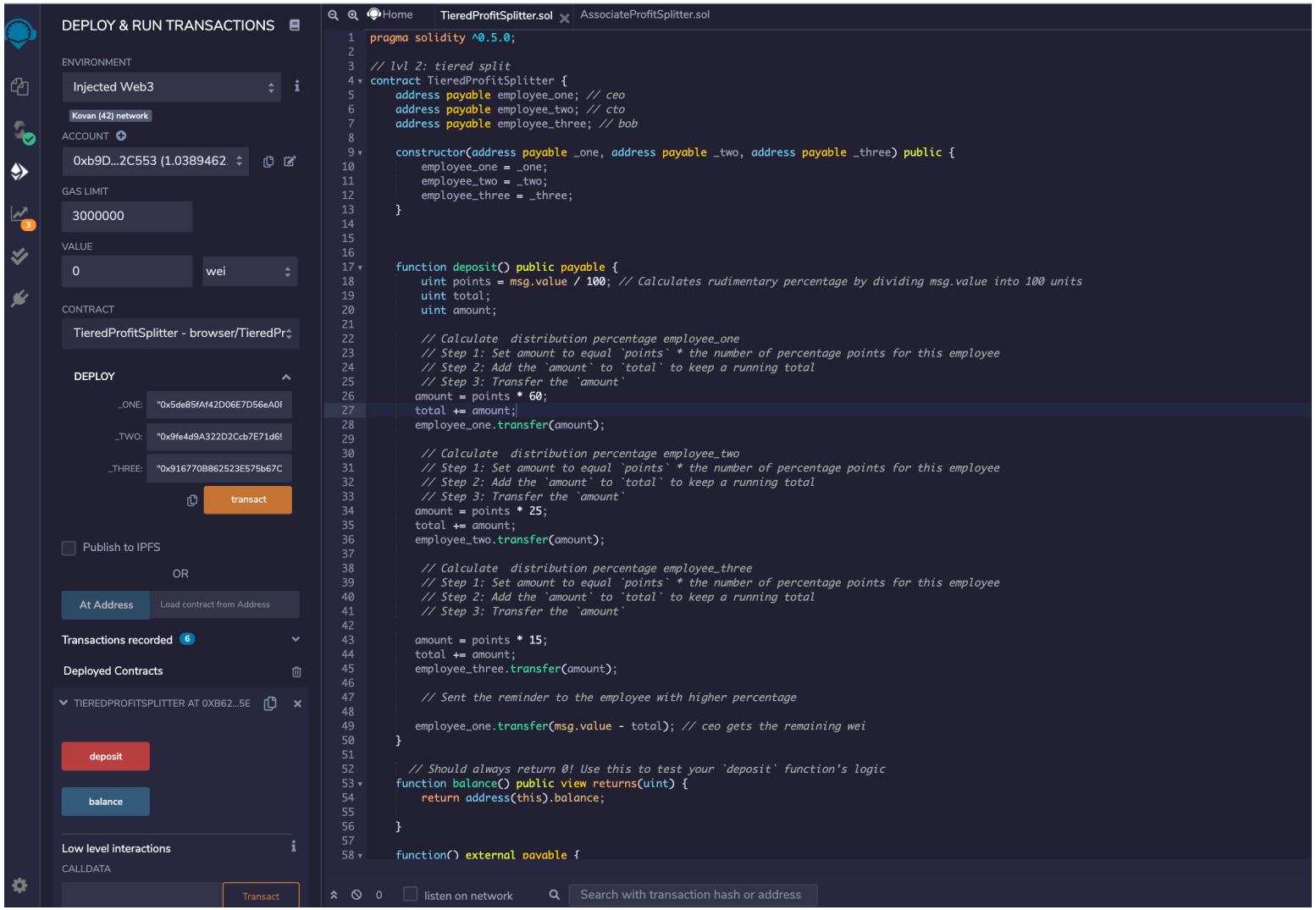
Test second contract



Deploy the the first contract in live Testnet

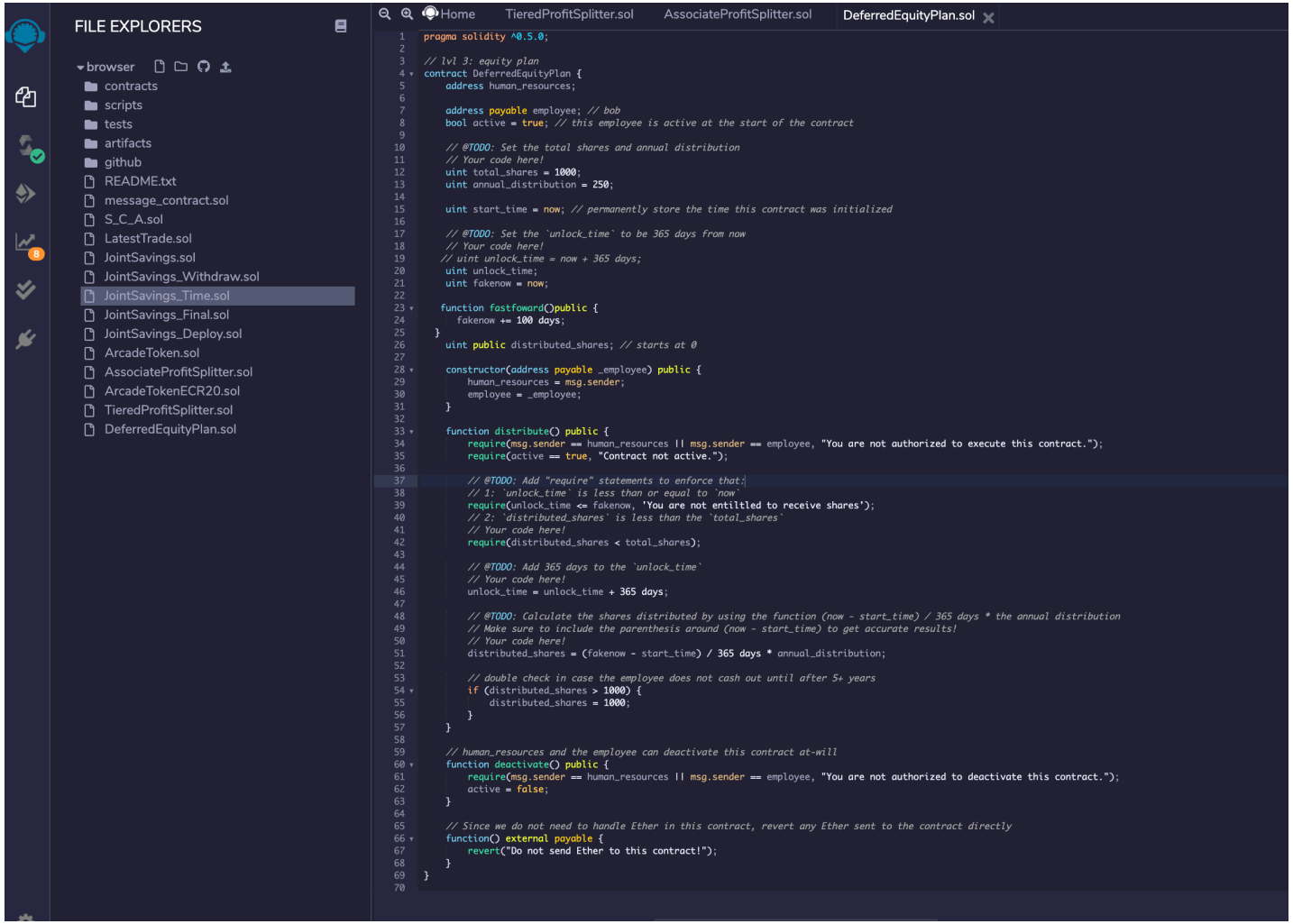


Deploy second contract in the live Testnet

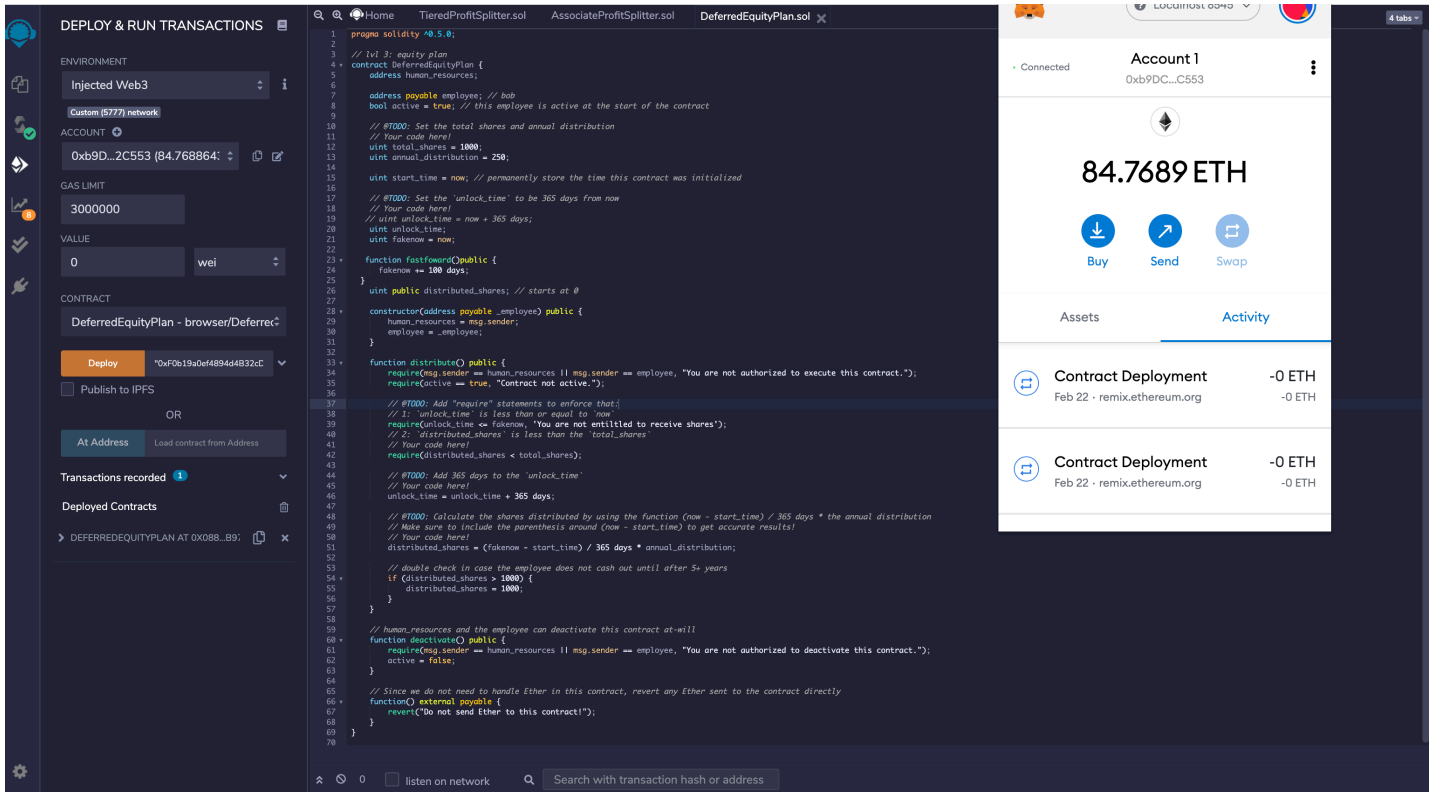



```
pragma solidity ^0.5.0;

// lvl 2: tiered split
contract TieredProfitSplitter {
    address payable employee_one; // ceo
    address payable employee_two; // cto
    address payable employee_three; // bob

    constructor(address payable _one, address payable _two, address payable _three) public {
        employee_one = _one;
        employee_two = _two;
        employee_three = _three;
    }



    function deposit() public payable {
        uint points = msg.value / 100; // Calculates rudimentary percentage by dividing msg.value into 100 units
        uint total;
        uint amount;

        // Calculate  distribution percentage employee_one
        // Step 1: Set amount to equal `points` * the number of percentage points for this employee
        // Step 2: Add the `amount` to `total` to keep a running total
        // Step 3: Transfer the `amount`
      amount = points * 60;
      total += amount;
      employee_one.transfer(amount);

        // Calculate  distribution percentage employee_two
        // Step 1: Set amount to equal `points` * the number of percentage points for this employee
        // Step 2: Add the `amount` to `total` to keep a running total
        // Step 3: Transfer the `amount`
      amount = points * 25;
      total += amount;
      employee_two.transfer(amount);

        // Calculate  distribution percentage employee_three
        // Step 1: Set amount to equal `points` * the number of percentage points for this employee
        // Step 2: Add the `amount` to `total` to keep a running total
        // Step 3: Transfer the `amount`

      amount = points * 15;
      total += amount;
      employee_three.transfer(amount);

        // Sent the reminder to the employee with higher percentage

      employee_one.transfer(msg.value - total); // ceo gets the remaining wei
    }

     // Should always return 0! Use this to test your `deposit` function's logic
    function balance() public view returns(uint) {
        return address(this).balance;

    }

    function() external payable {
```

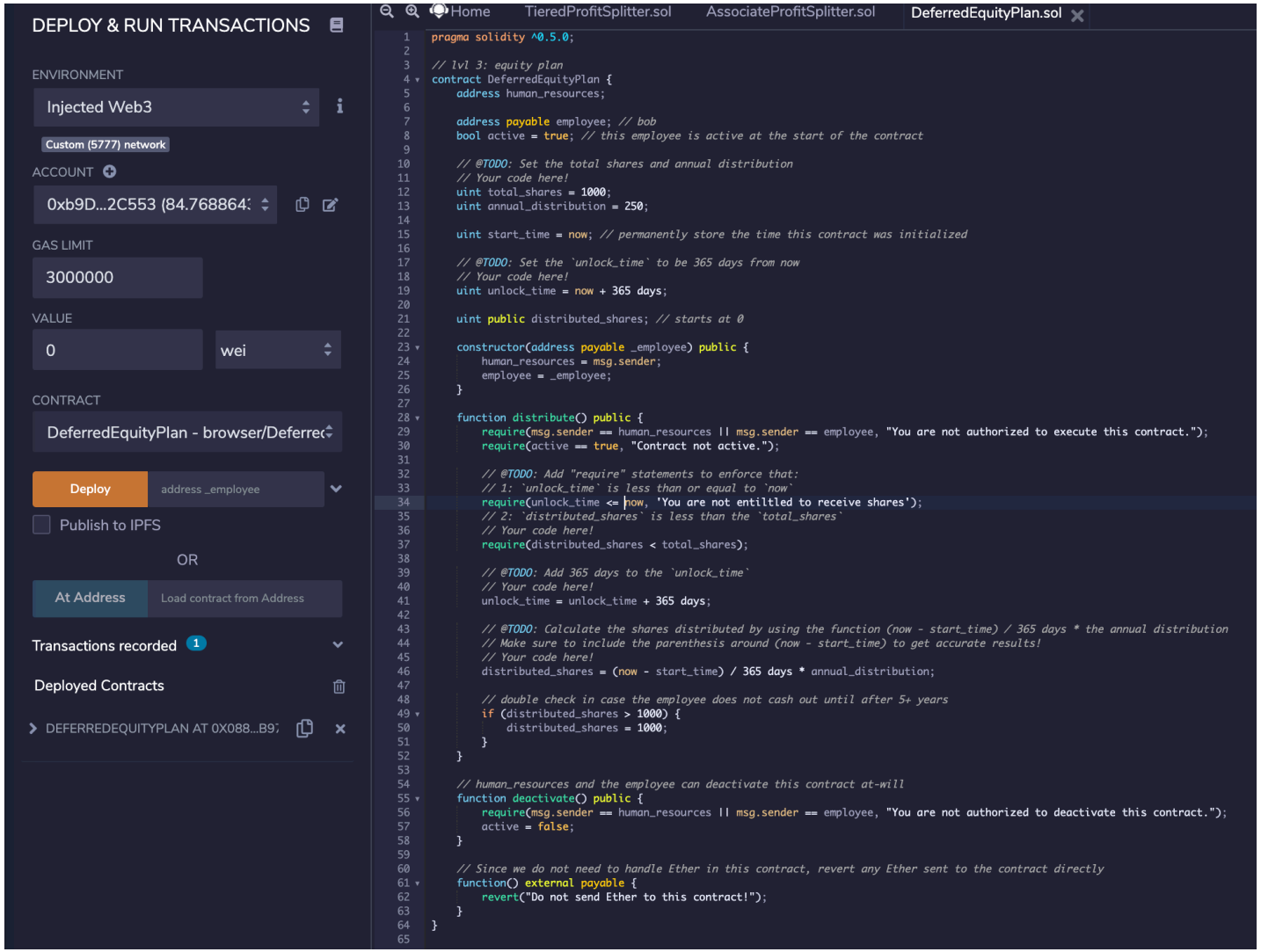
Create the third contract



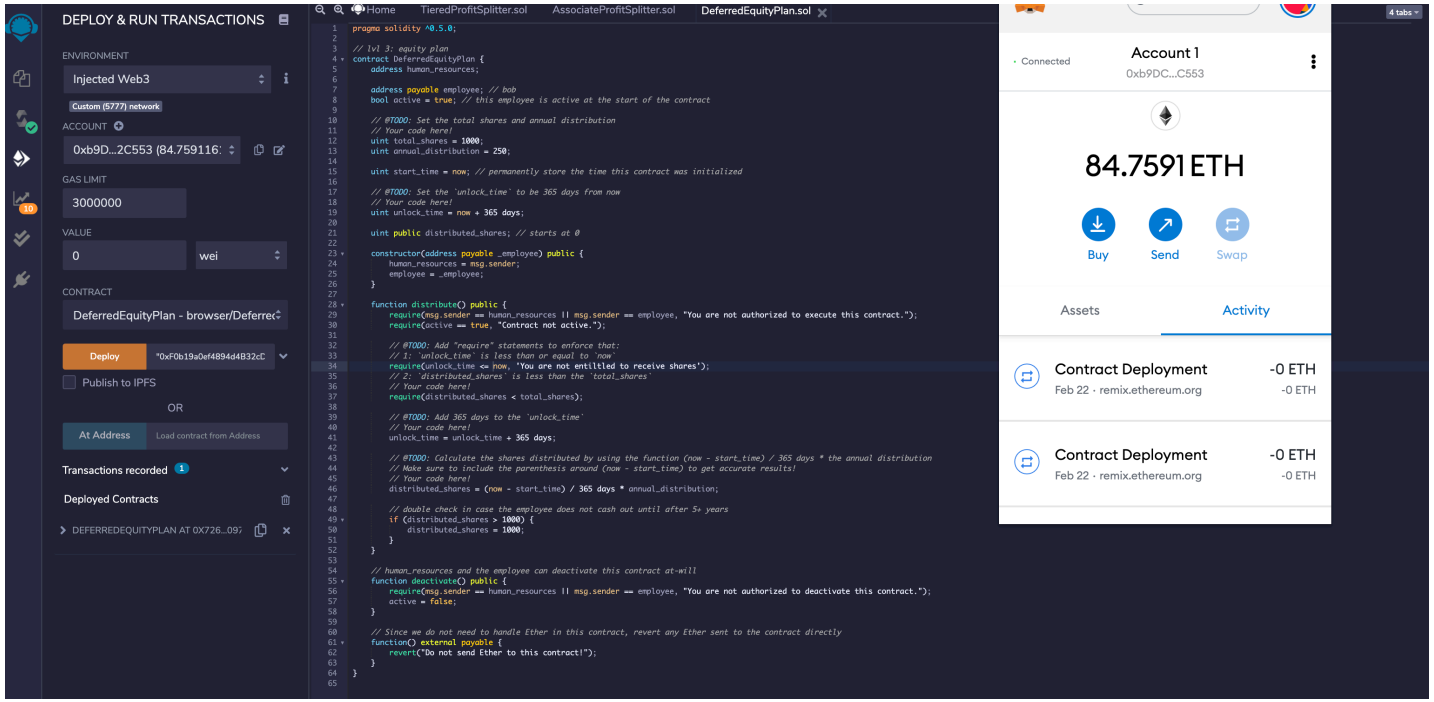
Compile the third contract



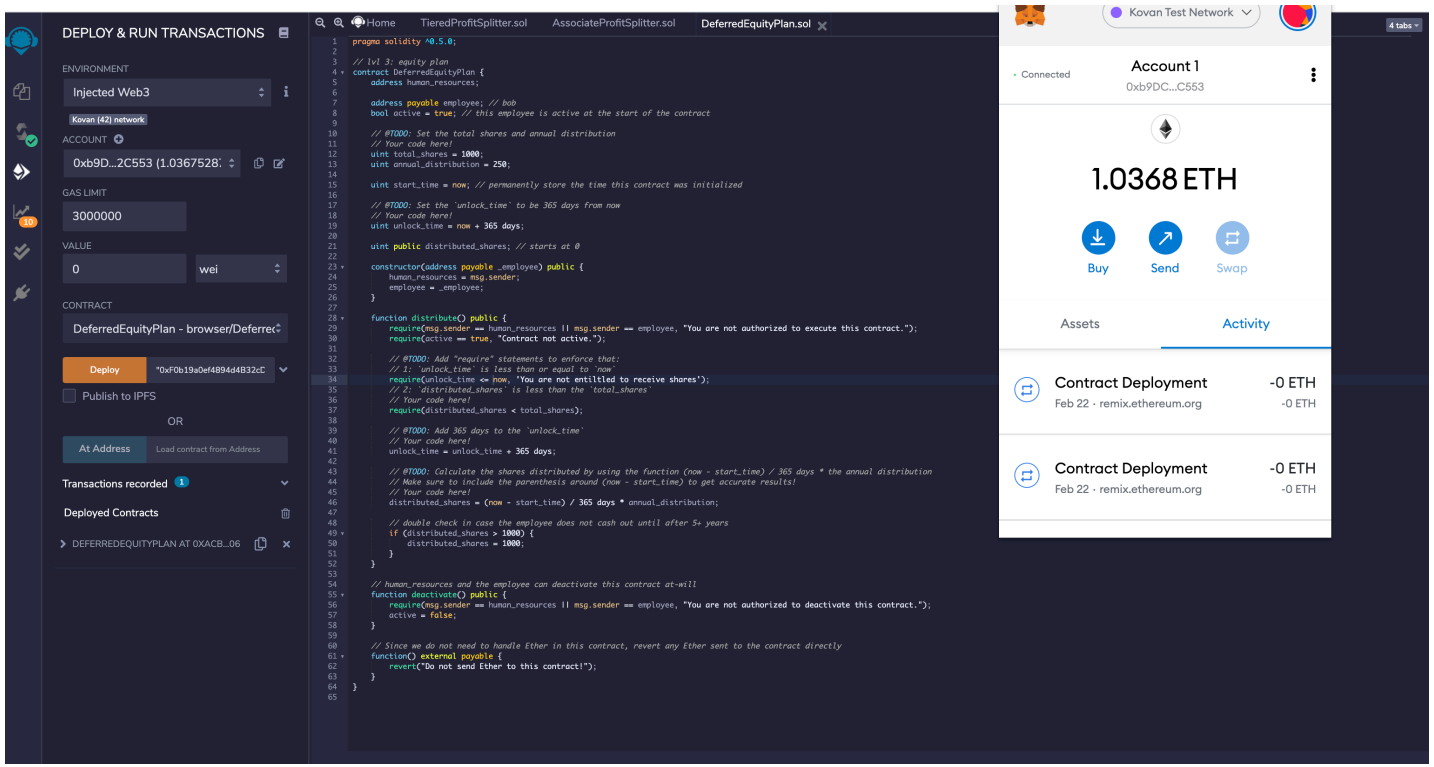
Deploy the third contract



Test the third contract



Deploy the Third Contract in live Tesnet



Copy the new addresses

0xB62DcFc3d9B5889cCbAf01d15a2EB81befF5EeE3

0xB62DcFc3d9B5889cCbAf01d15a2EB81befF5EeE3

0xACbbc4b7faC243eB95f1a12dFAd11E736E10613f