Create the first contract

Home | AssociateProfitSplitter.sol | 2 tabs

```
        employee_one = _one;
        employee_two = _two;
        employee_three = _three;
    }

    function balance() public view returns(uint) {
        return address(this).balance;
    }

    function deposit() public payable {
        // @TODO: Split `msg.value` into three
        uint amount = msg.value/3; // Your code here!

        // @TODO: Transfer the amount to each employee
        // Your code here!
        employee_one.transfer(amount);
        employee_two.transfer(amount);
        employee_three.transfer(amount);



        // @TODO: take care of a potential remainder by sending back to HR (`msg.sender`)
        // Your code here!
        msg.sender.transfer(msg.value-amount*3);
    }

    function() external payable {
        // @TODO: Enforce that the `deposit` function is called in the fallback function!
        // Your code here!
        deposit();
    }
}
```

Compiling first contract

SOLIDITY COMPILER

COMPILER

0.5.17+commit.d19bba13

Include nightly builds

LANGUAGE

Solidity

EVM VERSION

compiler default

COMPILER CONFIGURATION

Auto compile

Enable optimization 200

Hide warnings

Compile AssociateProfitSplitter.sol

CONTRACT

AssociateProfitSplitter (AssociateProfit

Publish on Swarm

Publish on Ipfs

Compilation Details

Home AssociateProfitSplitter.sol 2 tabs

```
        employee_two = _two;
        employee_three = _three;
    }

    function balance() public view returns(uint) {
        return address(this).balance;
    }

    function deposit() public payable {
        // @TODO: Split `msg.value` into three
        uint amount = msg.value/3; // Your code here!

        // @TODO: Transfer the amount to each employee
        // Your code here!
        employee_one.transfer(amount);
        employee_two.transfer(amount);
        employee_three.transfer(amount);



        // @TODO: take care of a potential remainder by sending back to
        // Your code here!
        msg.sender.transfer(msg.value-amount*3);
    }

    function() external payable {
        // @TODO: Enforce that the `deposit` function is called in the
        // Your code here!
        deposit();
    }
```

0 listen on network Search with transaction hash or address

```
remix.help(): Display this help message


 - Welcome to Remix 0.10.10 -


You can use this terminal to:
• Check transactions details and start debugging.
```

Deploy the first contract

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT

Injected Web3

Custom (5777) network

ACCOUNT

0xb9D...2C553 (96.825442:

GAS LIMIT

3000000

VALUE

0 wei

CONTRACT

AssociateProfitSplitter - browser/Asso

Deploy address _one, address _two,

Publish to IPFS

OR

At Address Load contract from Address

Transactions recorded 0

Deployed Contracts

Currently you have no contract instances to interact with.

Home AssociateProfitSplitter.sol 2 tabs

```
        employee_one = _one;
        employee_two = _two;
        employee_three = _three;
    }

    function balance() public view returns(uint) {
        return address(this).balance;
    }

    function deposit() public payable {
        // @TODO: Split `msg.value` into three
        uint amount = msg.value/3; // Your code here!

        // @TODO: Transfer the amount to each employee
        // Your code here!
        employee_one.transfer(amount);
        employee_two.transfer(amount);
        employee_three.transfer(amount);



        // @TODO: take care of a potential remainder by sending back to
        // Your code here!
        msg.sender.transfer(msg.value-amount*3);
    }

    function() external payable {
        // @TODO: Enforce that the `deposit` function is called in the
        // Your code here!
        deposit();
    }
```

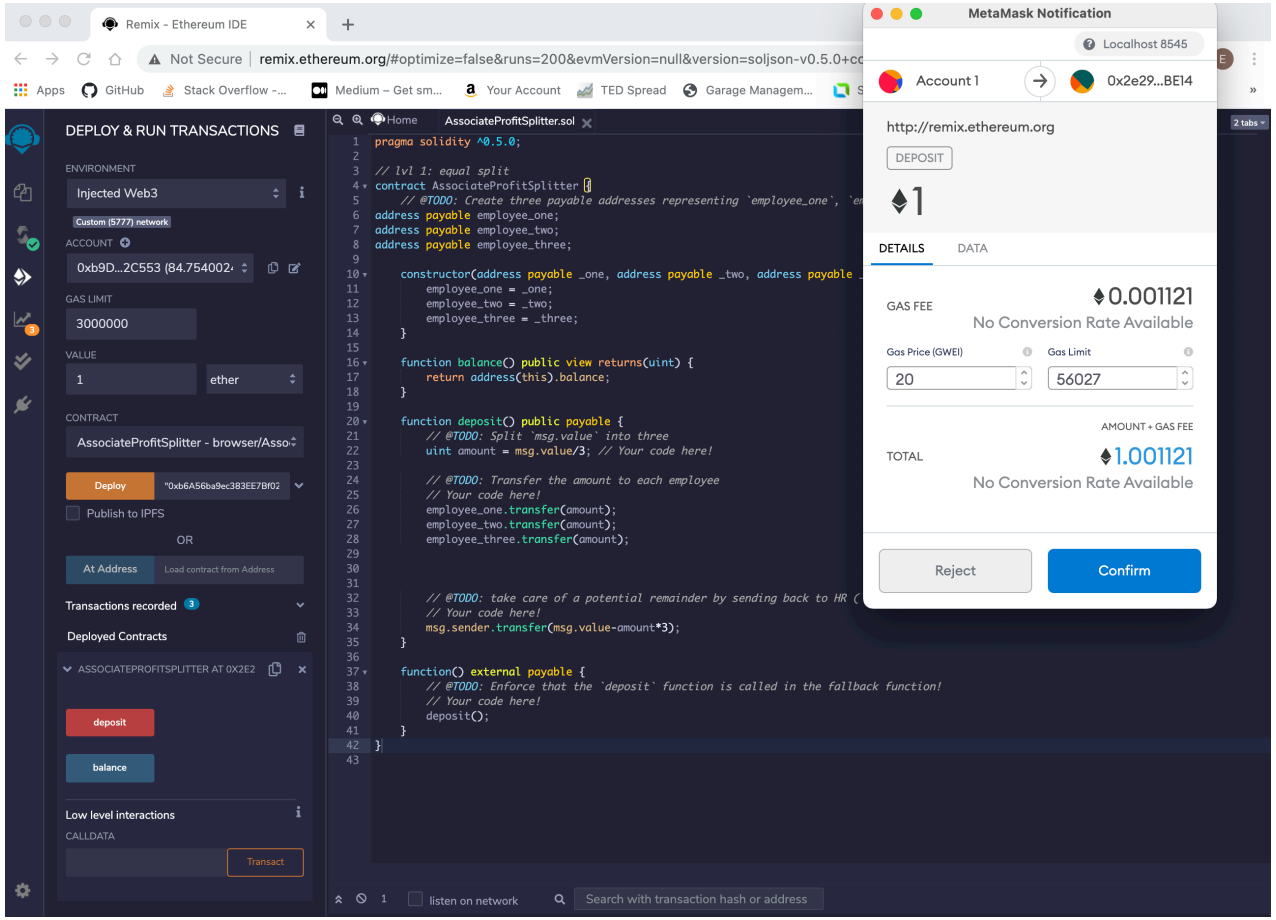
0 listen on network Search with transaction hash or address

```
remix.help(): Display this help message


 - Welcome to Remix 0.10.10 -


You can use this terminal to:
• Check transactions details and start debugging.
• Execute JavaScript scripts:
```

Acount Addresses balance before the deposit

ADDRESS	BALANCE	TX COUNT	INDEX
0×4f40f3D0169a02D37035548B6cE82284A4C083De	100.00 ETH	0	1
0×19632B1e8811863aCaBfdC331F1F638368DE2318	100.00 ETH	0	2
0×F0b19a0ef4894d4B32cDEc0e8Bd7a42bD95225c3	100.00 ETH	0	3

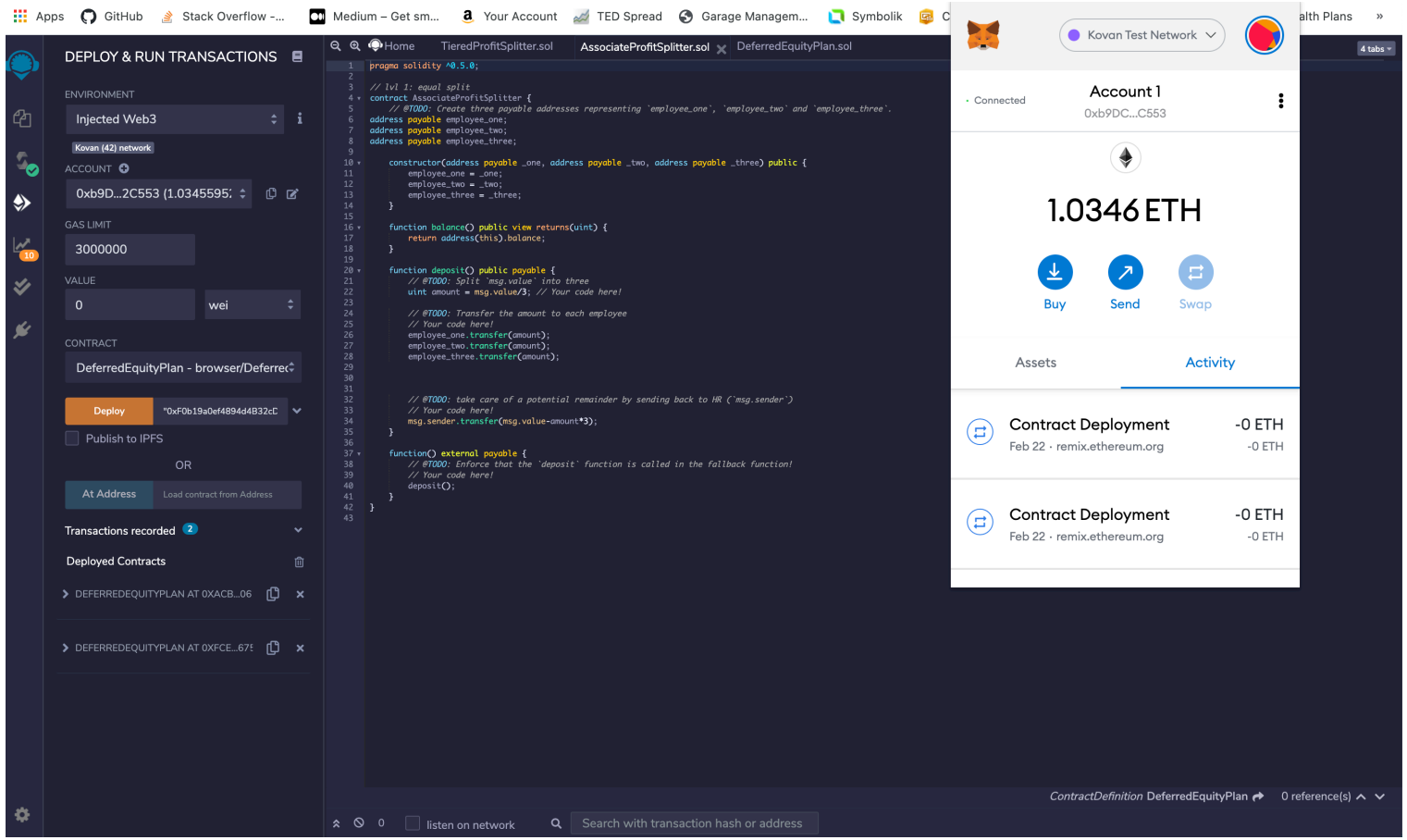
Deposit test



Accounts balance after the deposit

ADDRESS	BALANCE	TX COUNT	INDEX
0×b6A56ba9ec383EE7Bf02198CEB676FC20Fb322ae	100.33 ETH	0	7
0×f43ade89C92dCcC9842C6F423432D0e75d3daEAc	100.33 ETH	0	8
0×8BB420c15514Ac9e5f3abBe751ff9C4E8fC5FcCA	100.33 ETH	0	9

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

SOLIDITY COMPILER

COMPILER

0.5.0+commit.1d4f565a

Include nightly builds

LANGUAGE

Solidity

EVM VERSION

compiler default

COMPILER CONFIGURATION

Auto compile

Enable optimization 200

Hide warnings

Compile TieredProfitSplitter.sol

CONTRACT

TieredProfitSplitter (TieredProfitSplitte

Publish on Swarm

Publish on Ipfs

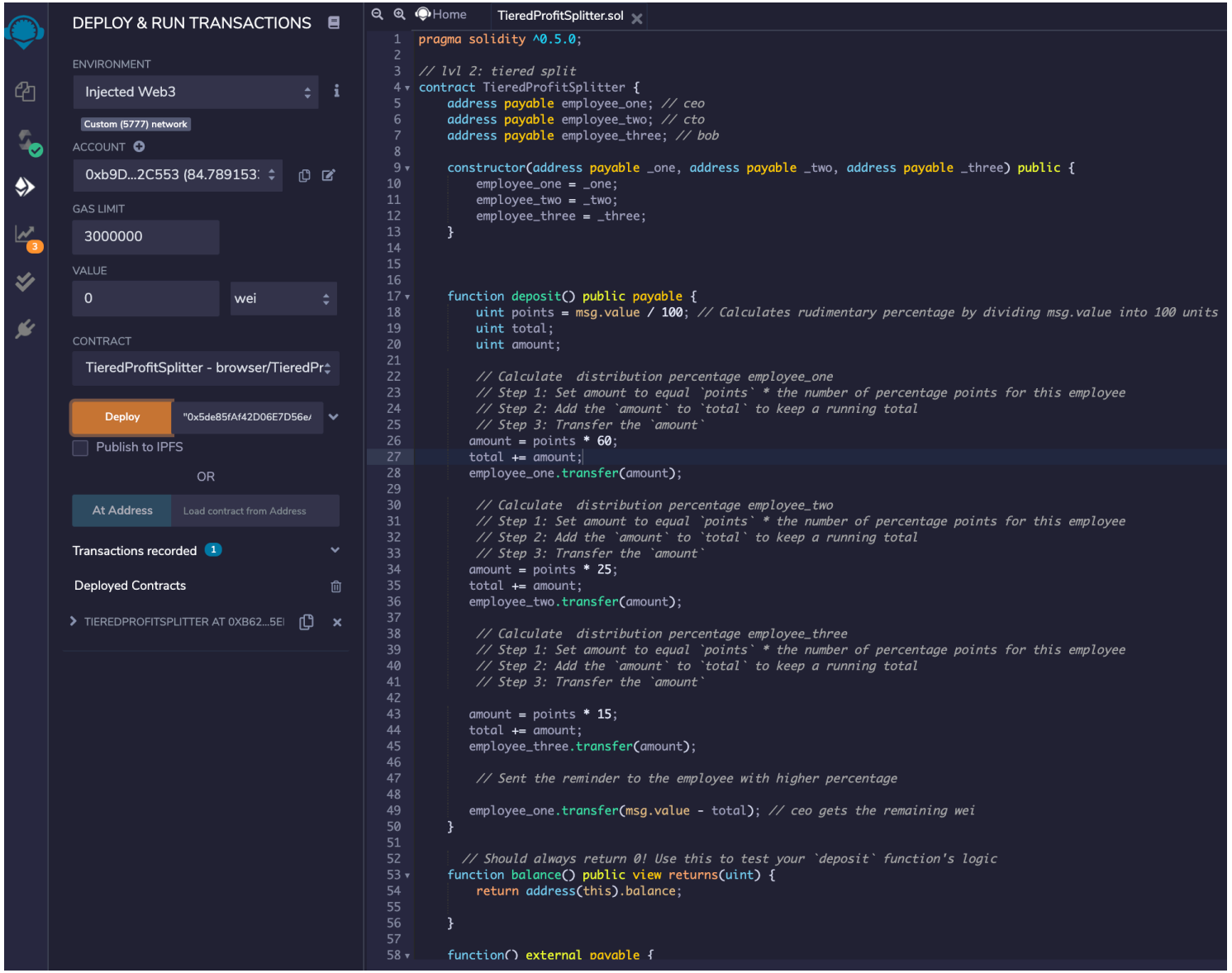
Compilation Details

ABI Bytecode

Home TieredProfitSplitter.sol

```
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();
    }
```

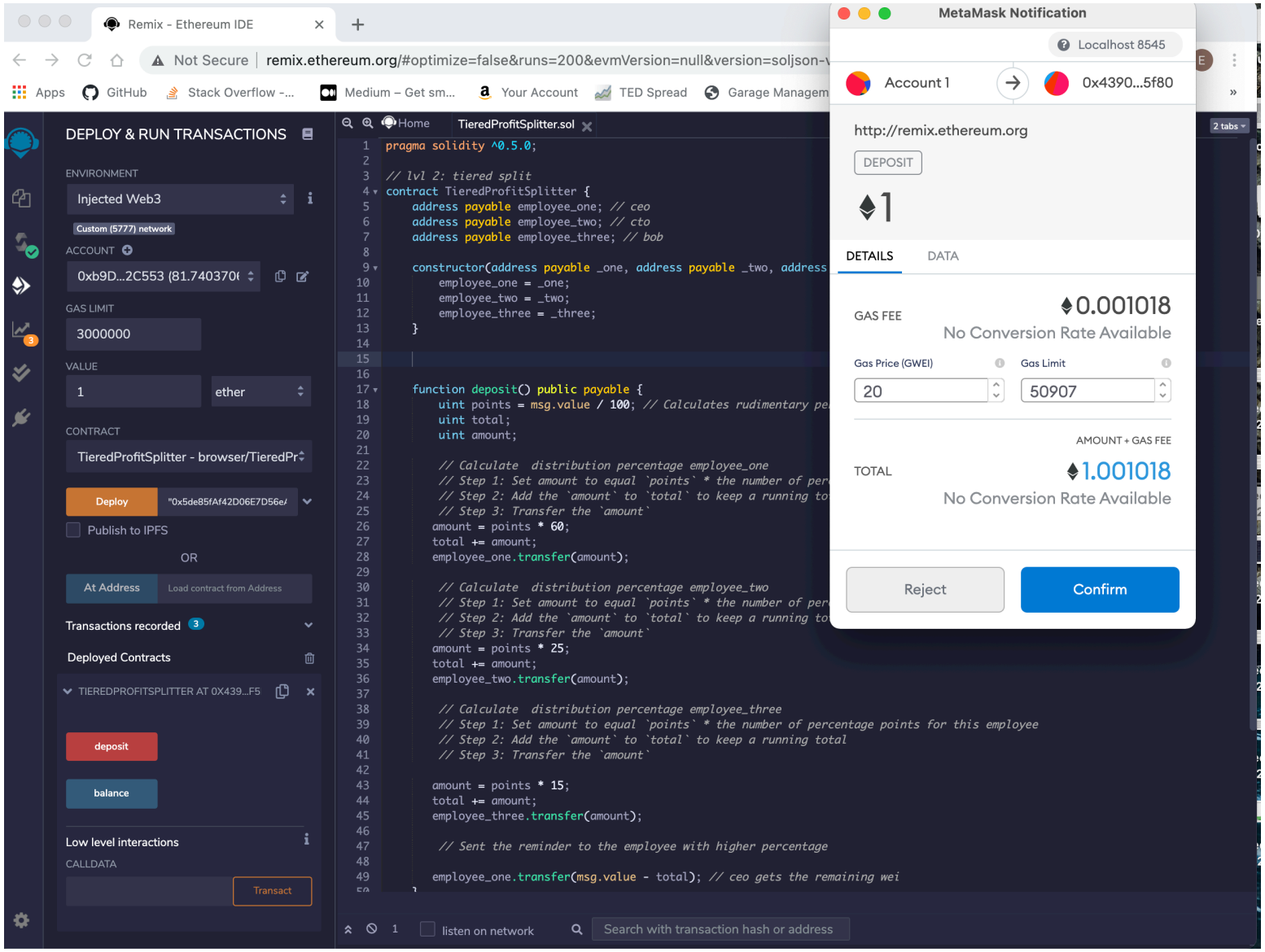
Deploy Second contract



Account Adresses before the deposit

ADDRESS	BALANCE	TX COUNT	INDEX
0×5de85fAf42D06E7D56eA0FeF469dd8BeF94Ed7D8	104.00 ETH	0	4
0×9fe4d9A322D2Ccb7E71d690dB01452f3116C7183	104.00 ETH	0	5
0×916770B862523E575b67C9aFBBE8420b8B3c14e5	104.00 ETH	0	6

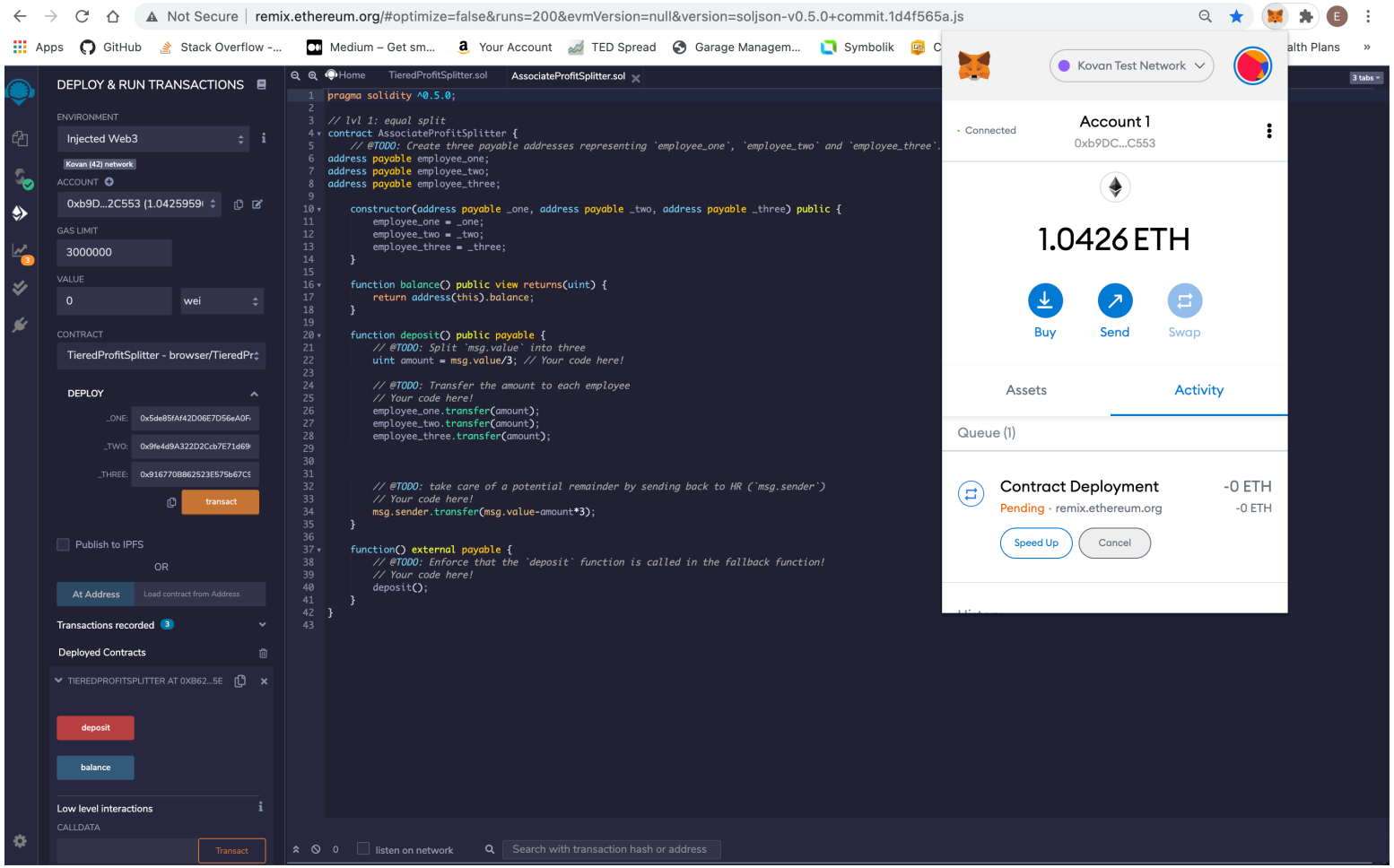
Test second contract



Account addresses after the deposit

ADDRESS	BALANCE	TX COUNT	INDEX
0×5de85fAf42D06E7D56eA0FeF469dd8BeF94Ed7D8	104.60 ETH	0	4
0×9fe4d9A322D2Ccb7E71d690dB01452f3116C7183	104.25 ETH	0	5
0×916770B862523E575b67C9aFBBE8420b8B3c14e5	104.15 ETH	0	6

Deploy the the first contract in live Testnet



Deploy second contract in the live Testnet

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT

Injected Web3

Kovan (42) network

ACCOUNT

0xb9D...2C553 (1.0389462

GAS LIMIT

3000000

VALUE

0 wei

CONTRACT

TieredProfitSplitter - browser/TieredPr

DEPLOY

_ONE: "0x5de85fAf42D06E7D56eA0f

_TWO: "0x9fe4d9A322D2Ccb7E71d6

_THREE: "0x916770B862523E575b67C

transact

Publish to IPFS

OR

At Address | Load contract from Address

Transactions recorded 6

Deployed Contracts

TIEREDPROFITSPLITTER AT 0XB62...5E

deposit

balance

Low level interactions

CALLDATA

Transact

Home | TieredProfitSplitter.sol | AssociateProfitSplitter.sol

```
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 {
```

0 listen on network Search with transaction hash or address

Create the third contract

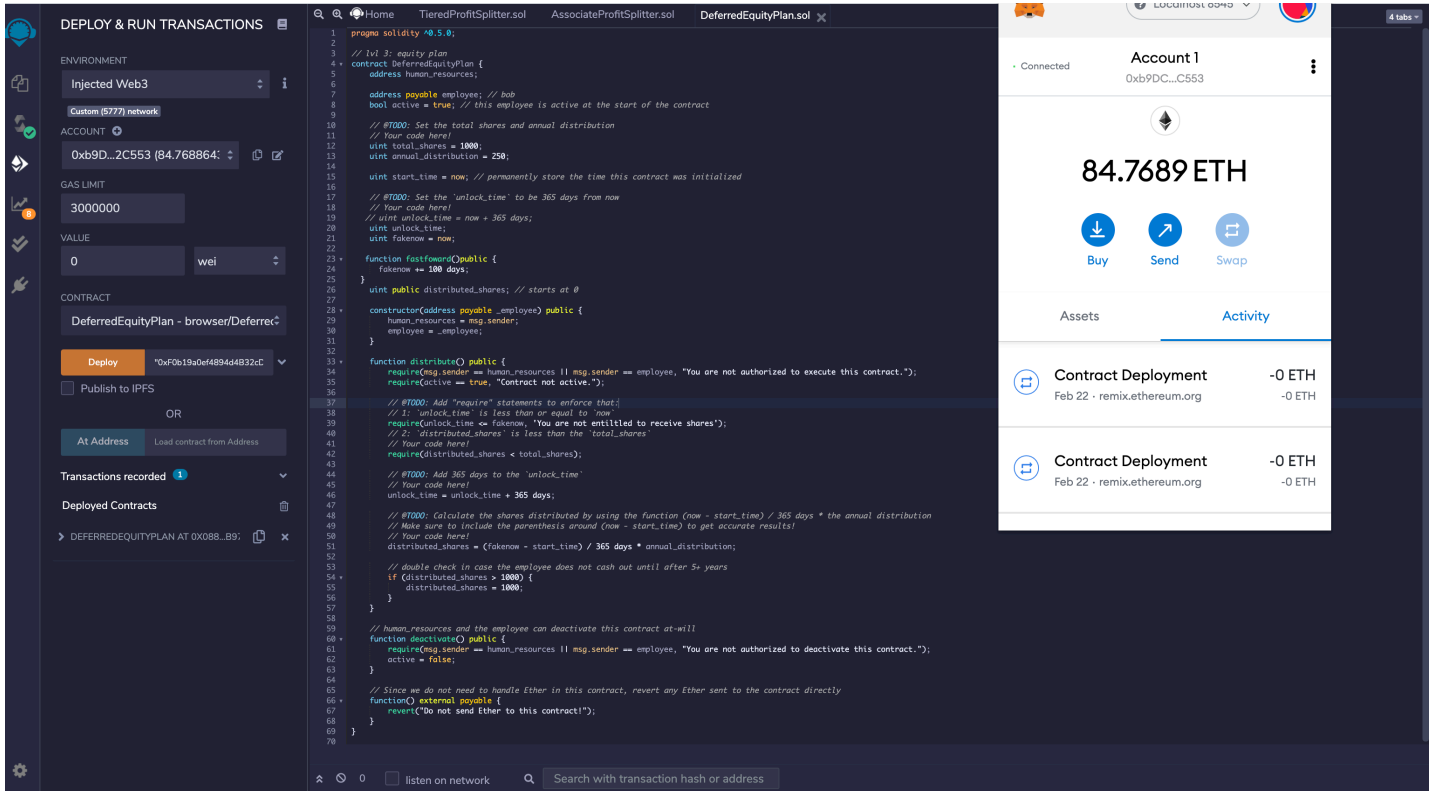
FILE EXPLORERS

- browser
  - contracts
  - scripts
  - tests
  - artifacts
  - github
  - README.txt
  - message_contract.sol
  - S_C_A.sol
  - LatestTrade.sol
  - JointSavings.sol
  - JointSavings_Withdraw.sol
  - JointSavings_Time.sol
  - JointSavings_Final.sol
  - JointSavings_Deploy.sol
  - ArcadeToken.sol
  - AssociateProfitSplitter.sol
  - ArcadeTokenECR20.sol
  - TieredProfitSplitter.sol
  - DeferredEquityPlan.sol

Home | TieredProfitSplitter.sol | AssociateProfitSplitter.sol | DeferredEquityPlan.sol

```
pragma solidity ^0.5.0;

// lvl 3: equity plan
contract DeferredEquityPlan {
    address human_resources;

    address payable employee; // bob
    bool active = true; // this employee is active at the start of the contract

    // @TODO: Set the total shares and annual distribution
    // Your code here!
    uint total_shares = 1000;
    uint annual_distribution = 250;

    uint start_time = now; // permanently store the time this contract was initialized

    // @TODO: Set the `unlock_time` to be 365 days from now
    // Your code here!
   // uint unlock_time = now + 365 days;
    uint unlock_time;
    uint fakenow = now;

   function fastfoward()public {
       fakenow += 100 days;
  }
    uint public distributed_shares; // starts at 0

    constructor(address payable _employee) public {
        human_resources = msg.sender;
        employee = _employee;
    }

    function distribute() public {
        require(msg.sender == human_resources || msg.sender == employee, "You are not authorized to execute this contract.");
        require(active == true, "Contract not active.");

        // @TODO: Add "require" statements to enforce that:
        // 1: `unlock_time` is less than or equal to `now`
        require(unlock_time <= fakenow, 'You are not entiltled to receive shares');
        // 2: `distributed_shares` is less than the `total_shares`
        // Your code here!
        require(distributed_shares < total_shares);

        // @TODO: Add 365 days to the `unlock_time`
        // Your code here!
        unlock_time = unlock_time + 365 days;

        // @TODO: Calculate the shares distributed by using the function (now - start_time) / 365 days * the annual distribution
        // Make sure to include the parenthesis around (now - start_time) to get accurate results!
        // Your code here!
        distributed_shares = (fakenow - start_time) / 365 days * annual_distribution;

        // double check in case the employee does not cash out until after 5+ years
        if (distributed_shares > 1000) {
            distributed_shares = 1000;
        }
    }

    // human_resources and the employee can deactivate this contract at-will
    function deactivate() public {
        require(msg.sender == human_resources || msg.sender == employee, "You are not authorized to deactivate this contract.");
        active = false;
    }

    // Since we do not need to handle Ether in this contract, revert any Ether sent to the contract directly
    function() external payable {
        revert("Do not send Ether to this contract!");
    }
}
```

Compile the third contract



Deploy the third contract

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT

Injected Web3

Custom (5777) network

ACCOUNT

0xb9D...2C553 (84.768864

GAS LIMIT

3000000

VALUE

0 wei

CONTRACT

DeferredEquityPlan - browser/Deferred

Deploy address _employee

Publish to IPFS

OR

At Address Load contract from Address

Transactions recorded 1

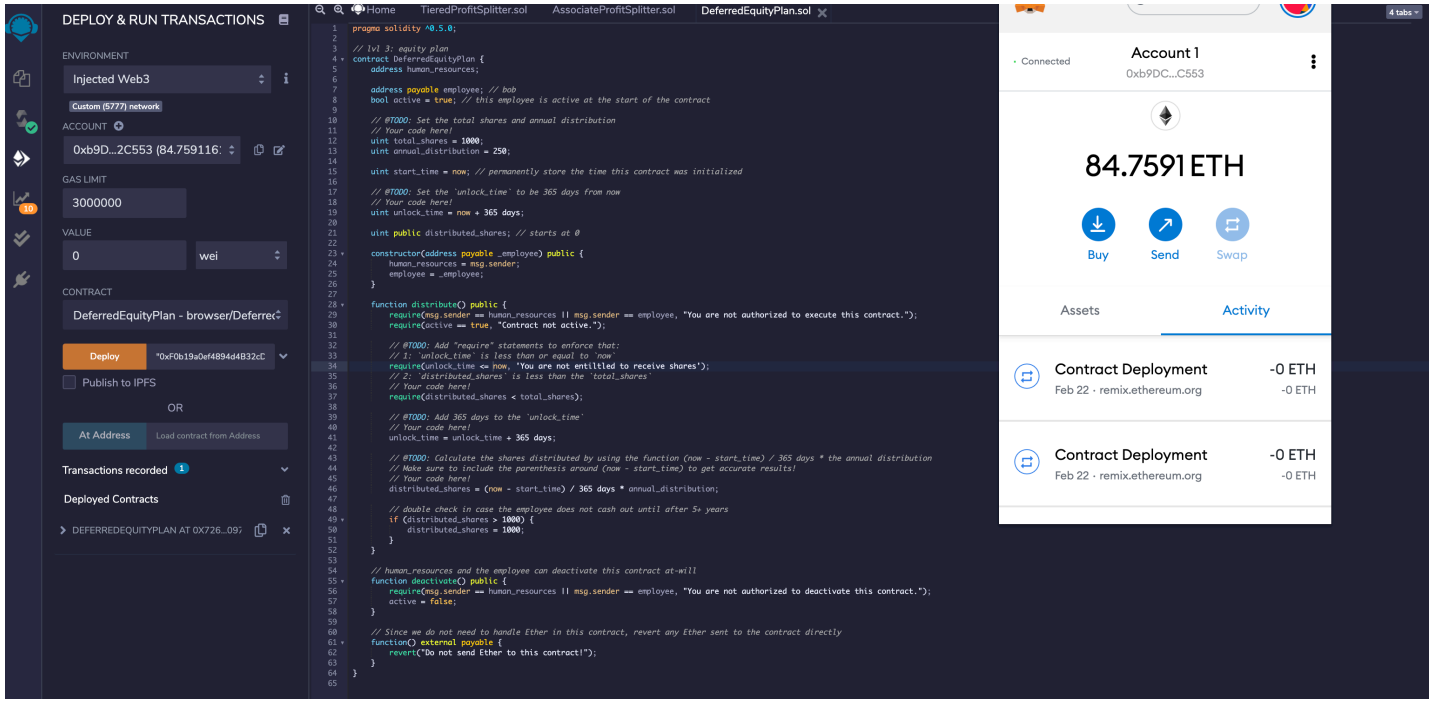
Deployed Contracts

DEFERREDEQUITYPLAN AT 0X088...B97

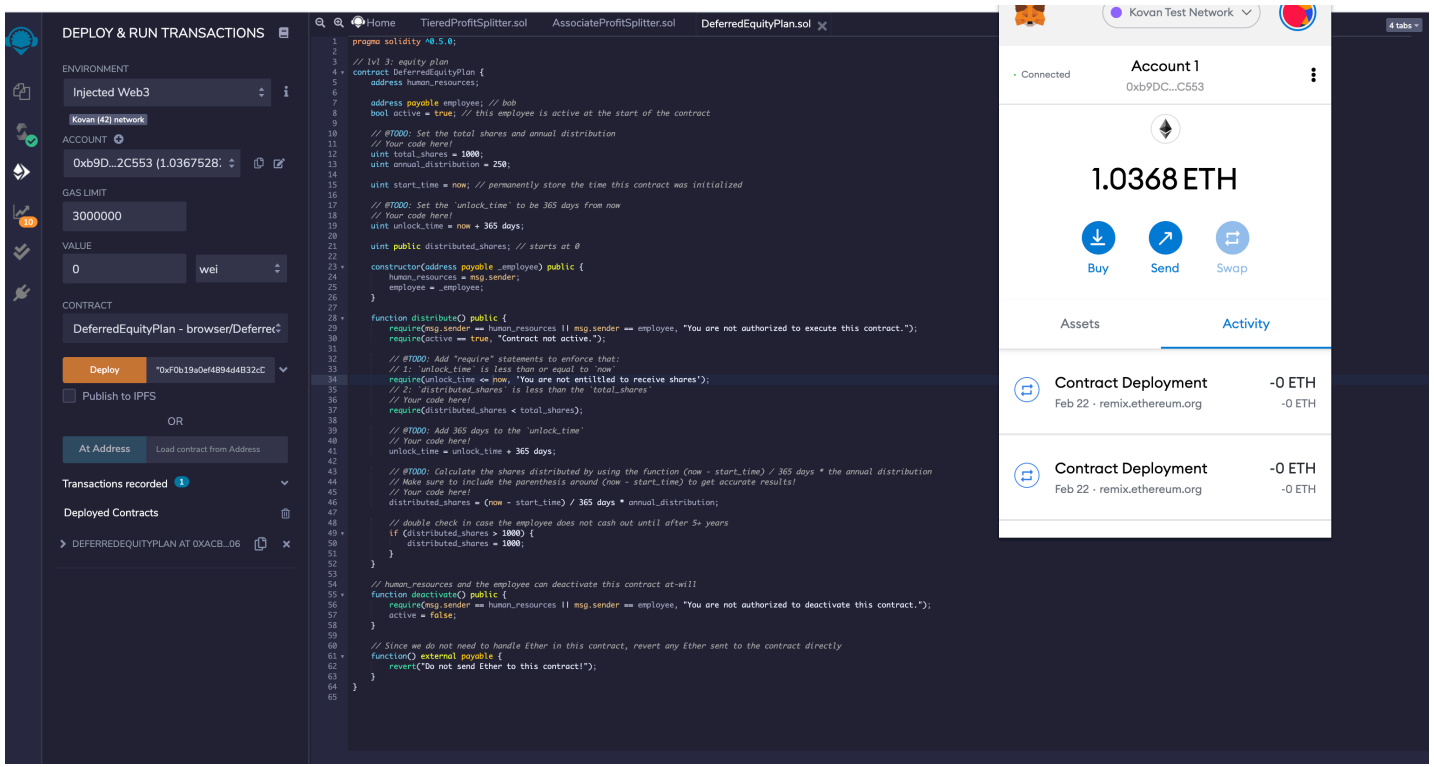
Home TieredProfitSplitter.sol AssociateProfitSplitter.sol DeferredEquityPlan.sol

```
pragma solidity ^0.5.0;

// lvl 3: equity plan
contract DeferredEquityPlan {
    address human_resources;

    address payable employee; // bob
    bool active = true; // this employee is active at the start of the contract

    // @TODO: Set the total shares and annual distribution
    // Your code here!
    uint total_shares = 1000;
    uint annual_distribution = 250;

    uint start_time = now; // permanently store the time this contract was initialized

    // @TODO: Set the `unlock_time` to be 365 days from now
    // Your code here!
    uint unlock_time = now + 365 days;

    uint public distributed_shares; // starts at 0

    constructor(address payable _employee) public {
        human_resources = msg.sender;
        employee = _employee;
    }

    function distribute() public {
        require(msg.sender == human_resources || msg.sender == employee, "You are not authorized to execute this contract.");
        require(active == true, "Contract not active.");

        // @TODO: Add "require" statements to enforce that:
        // 1: `unlock_time` is less than or equal to `now`
        require(unlock_time <= now, 'You are not entitled to receive shares');
        // 2: `distributed_shares` is less than the `total_shares`
        // Your code here!
        require(distributed_shares < total_shares);

        // @TODO: Add 365 days to the `unlock_time`
        // Your code here!
        unlock_time = unlock_time + 365 days;

        // @TODO: Calculate the shares distributed by using the function (now - start_time) / 365 days * the annual distribution
        // Make sure to include the parenthesis around (now - start_time) to get accurate results!
        // Your code here!
        distributed_shares = (now - start_time) / 365 days * annual_distribution;

        // double check in case the employee does not cash out until after 5+ years
        if (distributed_shares > 1000) {
            distributed_shares = 1000;
        }
    }

    // human_resources and the employee can deactivate this contract at-will
    function deactivate() public {
        require(msg.sender == human_resources || msg.sender == employee, "You are not authorized to deactivate this contract.");
        active = false;
    }

    // Since we do not need to handle Ether in this contract, revert any Ether sent to the contract directly
    function() external payable {
        revert("Do not send Ether to this contract!");
    }
}
```

Test the third contract



Deploy the Third Contract in live Tesnet



Copy the new addresses

0xB62DcFc3d9B5889cCbAf01d15a2EB81befF5EeE3

0xB62DcFc3d9B5889cCbAf01d15a2EB81befF5EeE3

0xACbbc4b7faC243eB95f1a12dFAd11E736E10613f