

## LAB 1: Deposit/Withdraw Ether

### 1.1 LAB: Smart contract Self Managing Funds

Pada lab ini anda akan mempelajari tentang bagaimana cara membuat smart contract yang akan mengatur keuangan anda. Anda akan mengirim Ether pada smart contract anda, lalu smart contract akan mengatur Ethernya dan dapat dikirim kepada siapapun. Ini seperti akun bank dengan code programming di dalamnya

Ini pun dapat digunakan sebagai escrow ether kedalam smart contract. Pertama yang kita perlukan adalah contoh setor dan Tarik tunai secara simple, lalu saya akan menunjukan kepada anda bagaimana smartcontract dapat mengunci sebuah dana menggunakan aktivasi waktu.

#### 1.1.1 Yang anda ketahui pada akhir lab ini

Mengetahui contract address dan sebuah global msg-object

Bagaimana smart contract mengatur pendanaan

Bagaimana untuk mengirim dan mengambil Ether dari dan ke smart contract

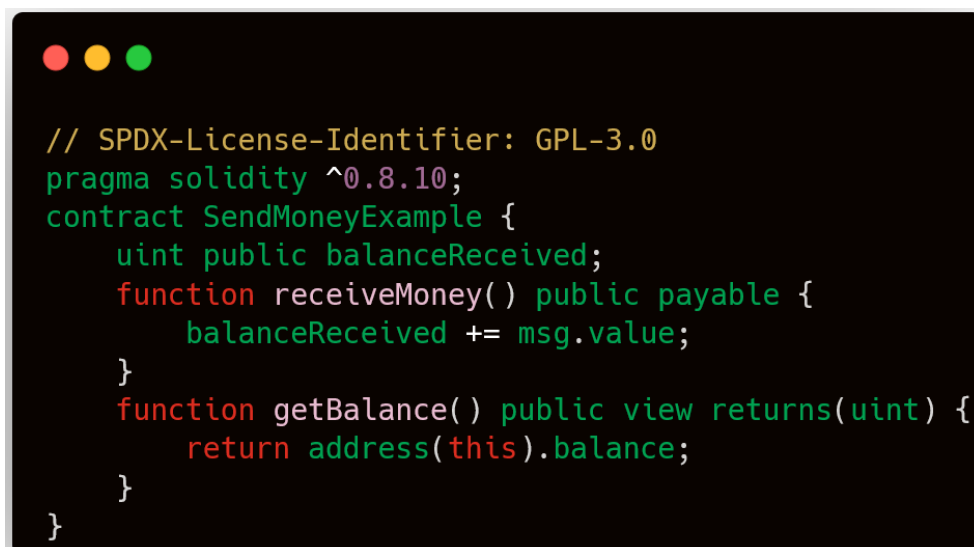
#### 1.1.2 Yang anda butuhkan

1. Browser
2. Koneksi koneksi
3. Waktu

### 1.2 Smart Contract

Mari kita buat smart contract yang simple dengan membuat file di remix dan paste kode dibawah ini

(SendMoneyExample.sol)



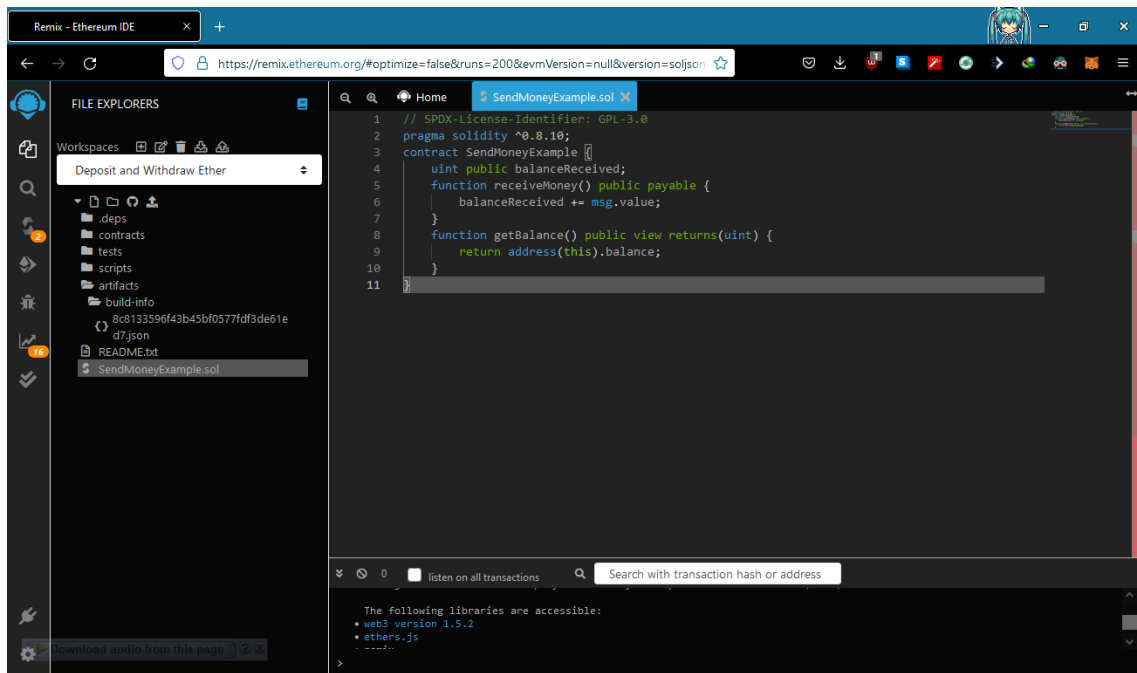
```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.10;
contract SendMoneyExample {
    uint public balanceReceived;
    function receiveMoney() public payable {
        balanceReceived += msg.value;
    }
    function getBalance() public view returns(uint) {
        return address(this).balance;
    }
}
```

uint public balanceReceived : sebuah variable public storage

balanceRecived += msg.value : msg-object adalah object global yang selalu ada dengan berisi beberapa informasi tentang transaksi yang berlangsung

function getBalance() public view returns(uint): view function adalah sebuah fungsi yang tidak alter terhadap penyimpanan (read-only) dan dapat mengembalikan informasi

address(this).balance : sabuah variable pada tipe alamat yang selalu mempunyai property yang disebut .balance yang dimana memberikan anda jumlah dari ether yang disimpan pada alamat tersebut.

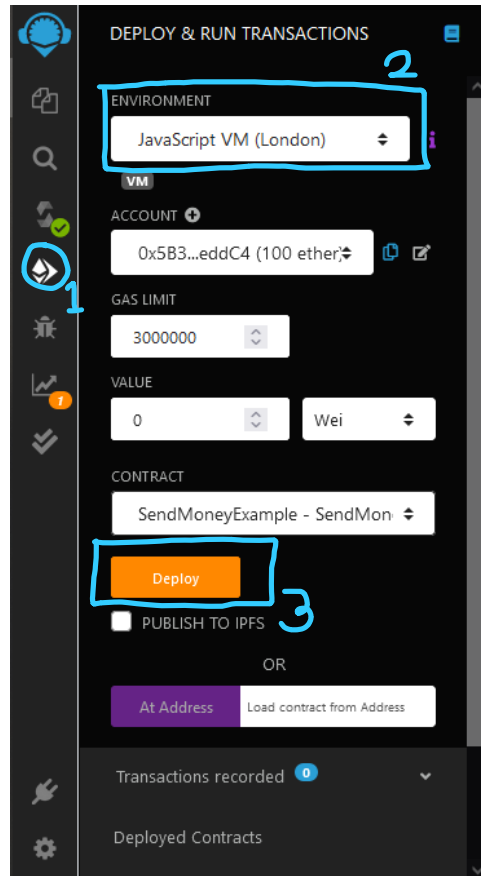


### 1.3 Mendeploy dan Menggunakan Smart Contract

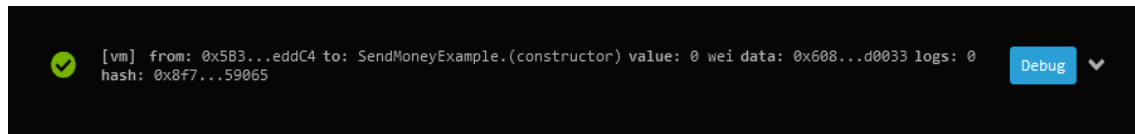
Pertama-tama kita harus mendeploy smart contract kita. Lalu kita dapat melihat jika kita dapat menyimpan ether dan mendapatkan balance ether kita dari smart contract.

#### 1.3.1 Mendeploy Smart Contract

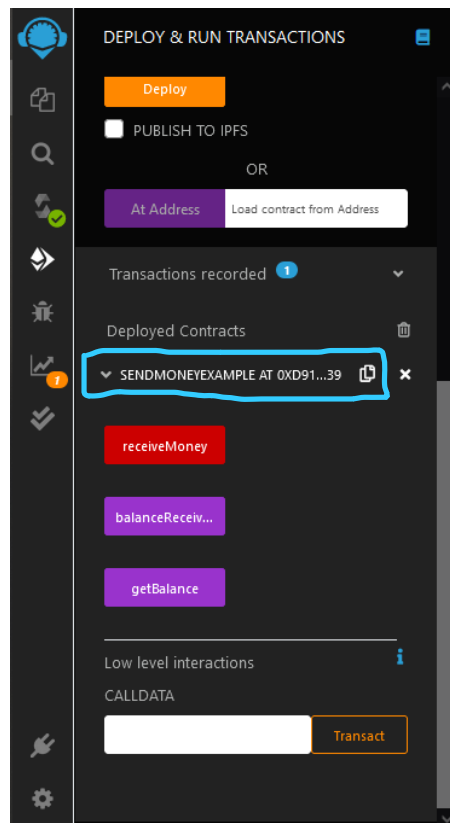
Klik ke Deploy and Run Transaction



Pada tab terminal anda dapat melihat status deploy smart contract kita apakah berhasil atau gagal

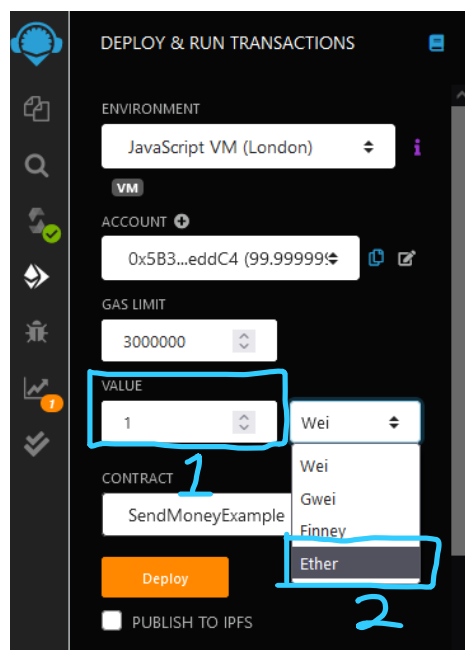


Untuk melihat smart contract yang telah terdeploy kita klik pada tab deployed contracts

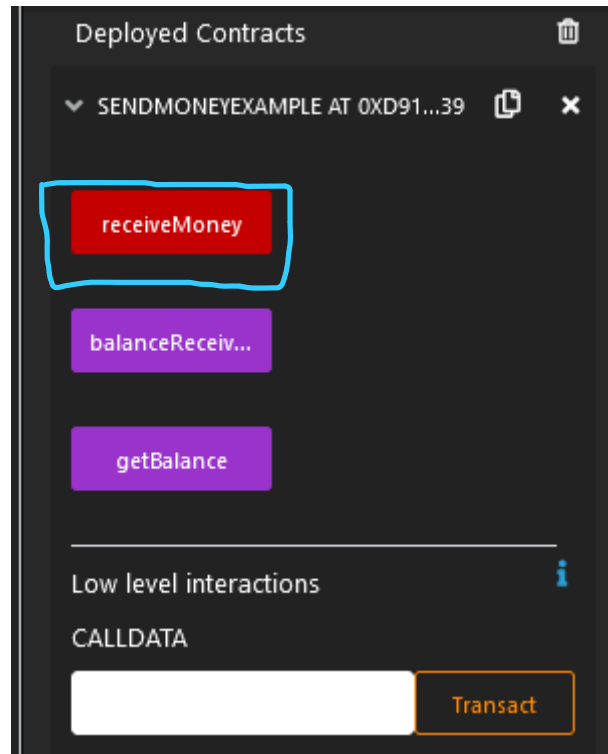


### 1.3.2 Mengirim Ether ke Smart Contract

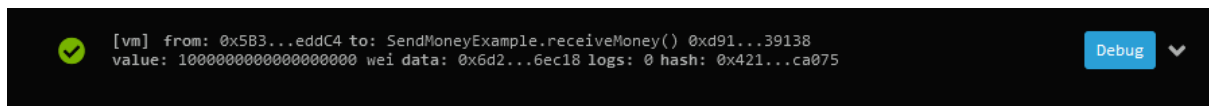
Untuk dapat mengirim ether ke dalam Smart Contract kita harus memasukan value yang akan kita kirim ke alamat Ethereum kita. Pertama scroll keatas dan kita dapat lihat value , lalu rubah nilai value menjadi 1 dan 'wei' menjadi ether.



Lalu scroll kebawah ke deployed contract dan kita dapat melihat dan klik tombol merah “receiveMoney”

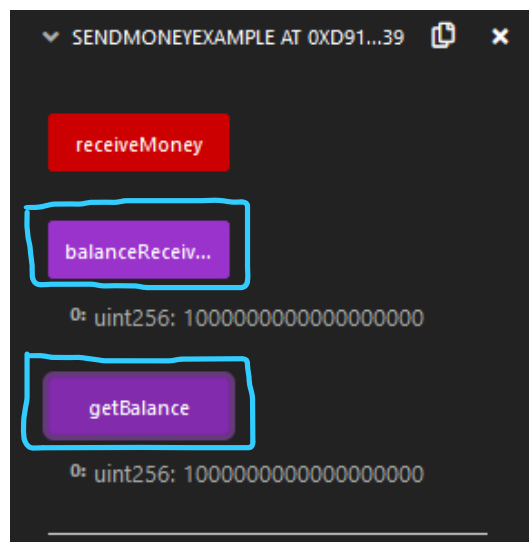


Pada kolom terminal kita dapat lihat keberhasilan dari transaksi kita



### 1.3.3 Cek Saldo di Smart Contract

Pada tahapan sebelumnya kita mengirim 1 Ether atau  $10^{18}$  Wei ke address kita, berdasarkan smartcontract yang kita telah buat sebelumnya, untuk mendapatkan nilai yang telah di terima kita dapat mengklik “balanceReceive” dan untuk melihat total saldo kita dapat mengklik “getBalance”



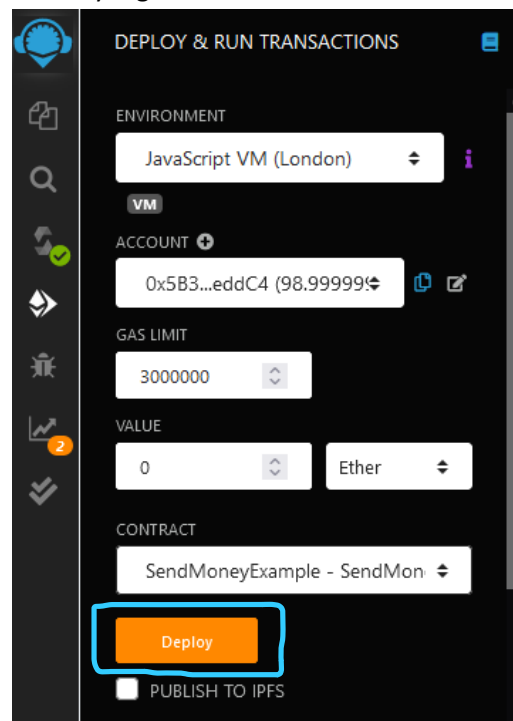
#### 1.4 Menarik Ether dari Smart Contract

Untuk dapat menarik saldo Ether dari address kita, maka tahapan selanjutnya adalah menambahkan line baru untuk fungsi penarikan ether. Tambahkan kode dibawah ini kedalam kode smartcontract yang sebelumnya kita buat.

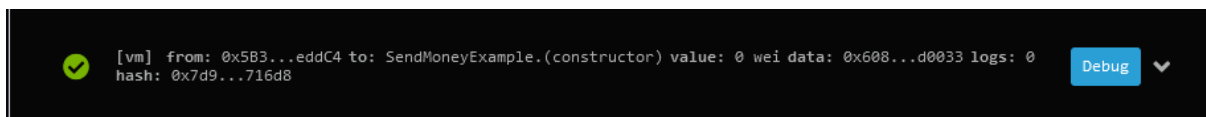
```
function withdrawMoney() public {  
    address payable to = payable(msg.sender);  
    to.transfer(getBalance());  
}
```

fungsi ini bertugas untuk mengirim semua saldo kita ke smart contract ke orang yang diberinama sebagai fungsi "withdrawMoney()"

##### 1.4.1 Deploy Smart Contract yang Terbaru



setelah smart contract kita terdeploy, Langkah selanjutnya adalah memastikan di terminal status deploy smart contract kita



lalu kita masukan lagi 1 Ether ke address kita seperti sebelumnya

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT  
JavaScript VM (London)

ACCOUNT  
0x5B3...eddC4 (99.99999)

GAS LIMIT  
3000000

VALUE  
1

CONTRACT  
SendMoneyExample

Wei  
Wei  
Gwei  
Finney  
Ether

Deploy

PUBLISH TO IPFS

[vm] from: 0x5B3...eddC4 to: SendMoneyExample.receiveMoney() 0xd91...39138  
value: 1000000000000000000 wei data: 0x6d2...6ec18 logs: 0 hash: 0x421...ca075

Debug

SENDMONEYEXAMPLE AT 0XD91...39

receiveMoney

withdrawMon...

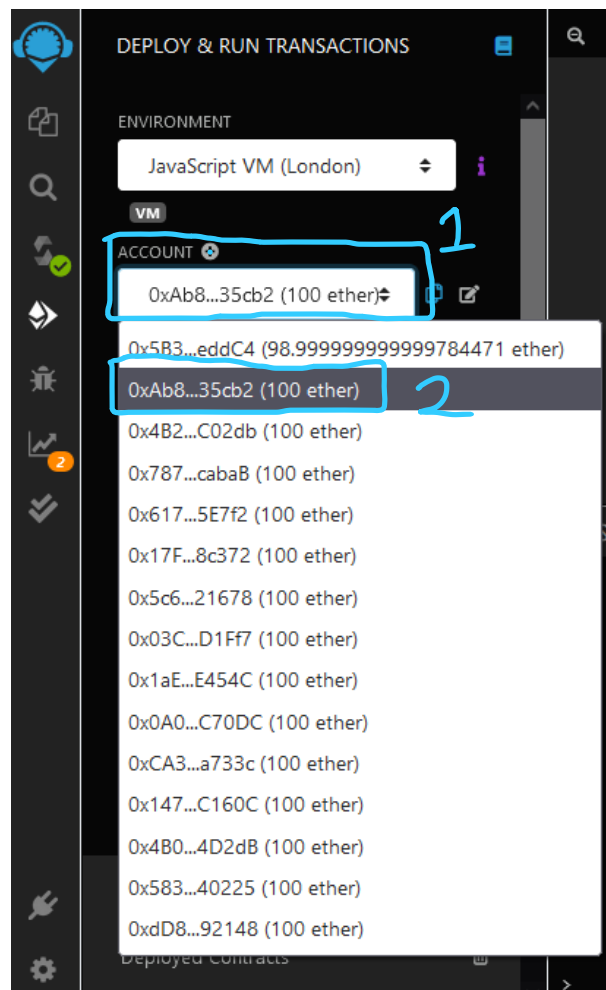
balanceReceiv...

getBalance

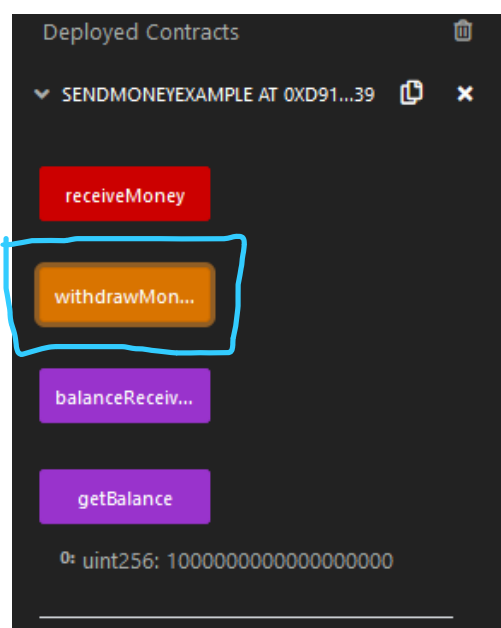
0: uint256: 1000000000000000000

#### 1.4.2 Menarik Saldo dari Smart Contract

Pada tahapan ini kita akan menarik saldo dari alamat akun yang berbeda



Lalu kita klik tombol withdrawMoney



The screenshot displays the 'DEPLOY & RUN TRANSACTIONS' interface in the Remix IDE. On the left sidebar, there are icons for environment selection, account selection, and transaction execution. The main panel has two sections: 'ENVIRONMENT' and 'ACCOUNT'.  
 - The 'ENVIRONMENT' section features a dropdown menu currently set to 'JavaScript VM (London)'.  
 - The 'ACCOUNT' section features a dropdown menu showing a list of accounts. The first account, '0xAb8...35cb2 (100.99999 ether)', is highlighted with a red rectangular box.  
 Below the highlighted account, several other accounts are listed, each followed by its balance in ether:  
 - 0x5B3...eddC4 (98.99999999999784471 ether)  
 - 0x4B2...C02db (100 ether)  
 - 0x787...cabaB (100 ether)  
 - 0x617...5E7f2 (100 ether)  
 - 0x17F...8c372 (100 ether)  
 - 0x5c6...21678 (100 ether)  
 - 0x03C...D1Ff7 (100 ether)  
 - 0x1aE...E454C (100 ether)  
 - 0x0A0...C70DC (100 ether)  
 - 0xCA3...a733c (100 ether)  
 - 0x147...C160C (100 ether)  
 - 0x4B0...4D2dB (100 ether)  
 - 0x583...40225 (100 ether)  
 - 0xdD8...92148 (100 ether)

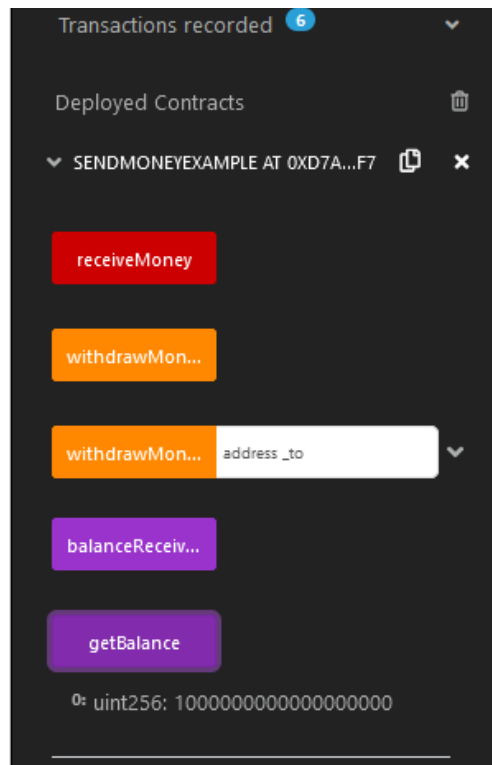
### 1.5 Menarik Saldo ke Akun Spesifik

```
function withdrawMoneyTo(address payable _to) public {
    _to.transfer(getBalance());
}
```



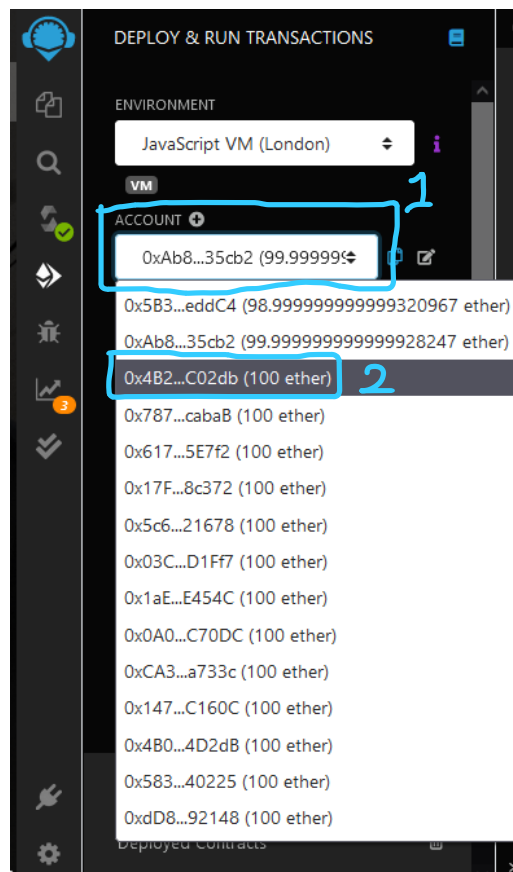
### 1.5.1 Deploy Kembali Smart Contract Kita

Deploy Kembali smart contract kita dan kirim satu ether ke akun dan cek Kembali apakah ether sudah di terima

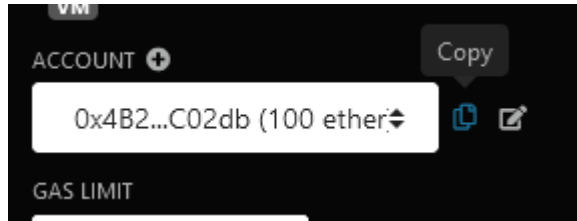


### 1.5.2 Test fungsi "withdrawMoneyTo"

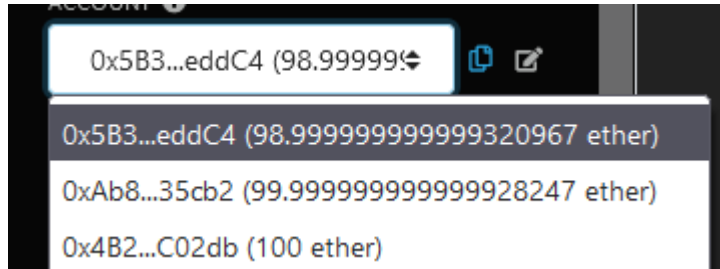
1. Gunakan akun yang ketiga pada list akun kita



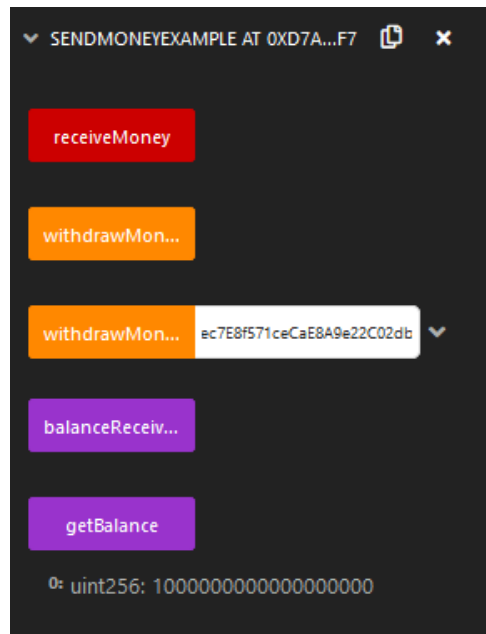
2. Klik tombol copy untuk menyalin address akun kita



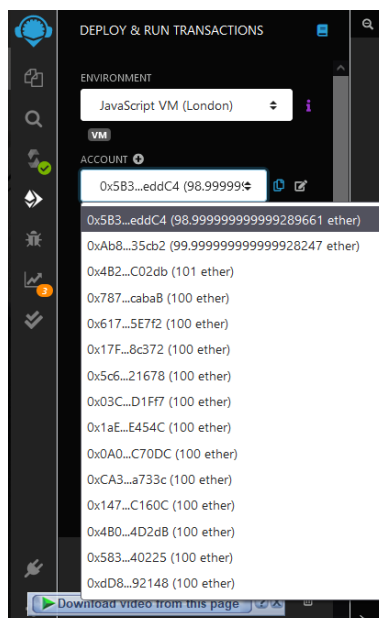
3. Kembali lagi ke akun pertama kita



4. Masukkan alamat akun yang kita telah salin sebelumnya ke kolom "withdrawMoneyTo" dan klik tombol "withdrawMoneyTo"



5. Lihat pada bagian akun dan lihat ada kejanggalan bahwa nilai ether yang diterima pas 101 ether



Mengapa bisa 101 ether? Dikarenakan kita mengirim dari akun 1 ke smartcontract, dengan menginstruksikan smartcontract untuk mengirimkan semua saldo ke alamat smartcontract ke akun ke tiga anda di list akun. Pembayaran fees dibayarkan oleh akun pertama jadinya akun ke 3 menerima 1 full ether

#### 1.6 Penguncian Pengambilan Saldo

Yang kita butuhkan adalah menyimpan "block.timestamp" di suatu tempat. Ada beberapa metode untuk melakukan ini, saya lebih suka membiarkan pengguna tahu berapa lama itu terkunci. Jadi, alih-alih menyimpan stempel waktu setoran, saya akan menyimpan stempel waktu yang terkunci. Ayo lihat Apa yang terjadi di sini:

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.10;
contract SendMoneyExample {
    uint public balanceReceived;
    uint public lockedUntil;
    function receiveMoney() public payable {
        balanceReceived += msg.value;
        lockedUntil = block.timestamp + 1 minutes;
    }
    function getBalance() public view returns(uint) {
        return address(this).balance;
    }
    function withdrawMoney() public {
        if(lockedUntil < block.timestamp) {
            address payable to = payable(msg.sender);
            to.transfer(getBalance());
        }
    }
    function withdrawMoneyTo(address payable _to) public {
        if(lockedUntil < block.timestamp) {
            _to.transfer(getBalance());
        }
    }
}
```

### 1.6.1 Deploy dan Test Smart Contract