Nama : Muhamad Fauzan Anshori

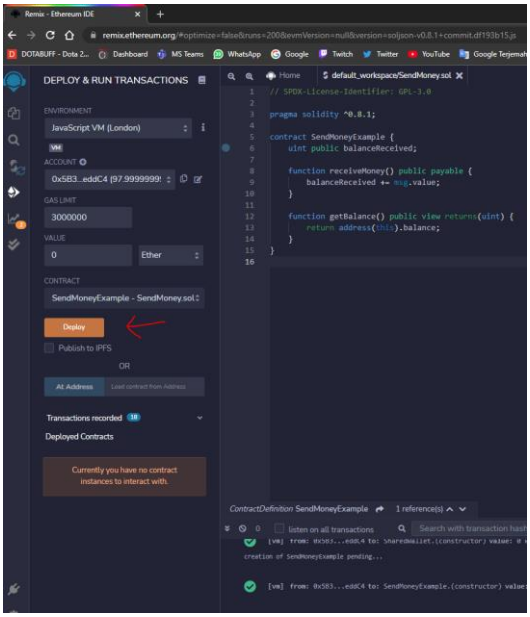
NIM : 1103190035

Kelas : TK-42-PIL

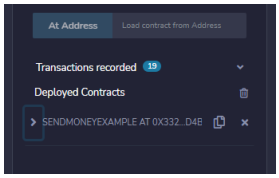
Dokumentasi Hands on Lab with Ethereum Developer Guide

I. LAB 1: Deposit/Withdraw Ether

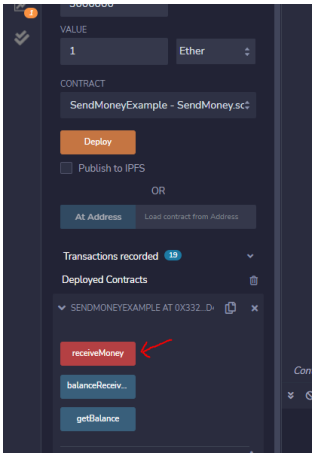
Membuat simple smart contract, Setelah itu jalankan dengan tombol deploy



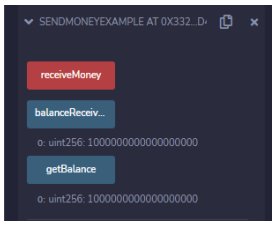
Contract yang telah dijalankan akan muncul di bagian bawah



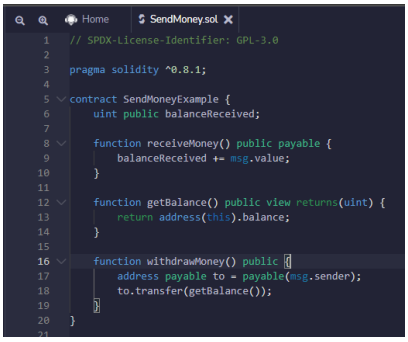
Klik contract tersebut dan mulai untuk menerima ether dengan mengisi value 1 ether, setelah itu tekan tombol receiveMoney



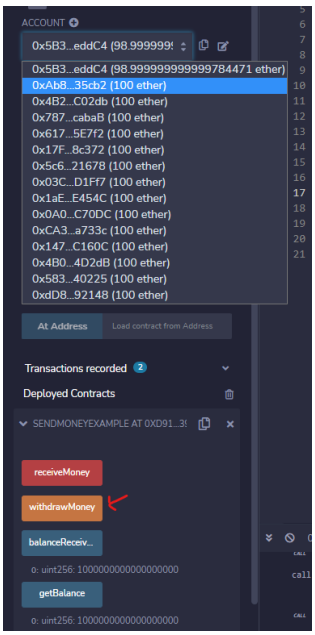
Tombol balanceReceived untuk melihat jumlah yang telah diterima, dan getBalance untuk melihat jumlah ether pada smart contract



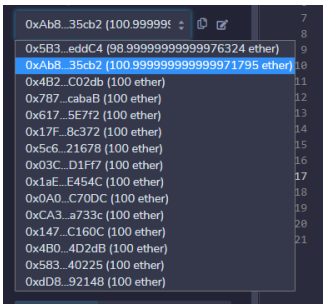
Menambahkan fungsi withdraw ke smart contract



Redeploy smart contract berisi 1 ether dan pilih akun withdraw setelah itu pilih box withdrawMoney()



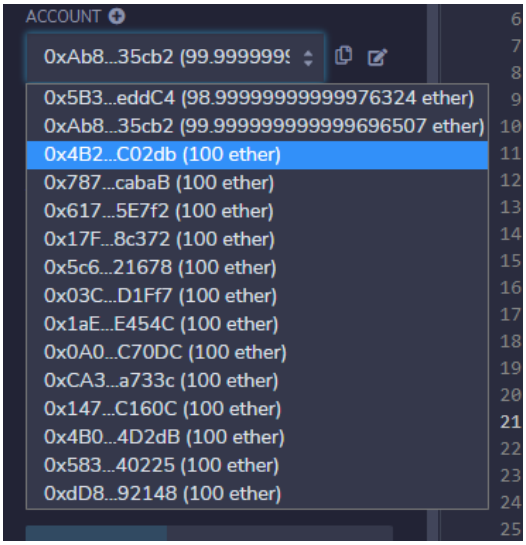
Dapat terlihat ether akun tersebut telah bertambah

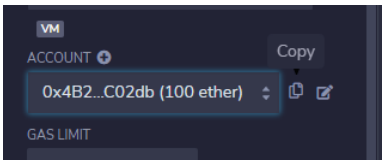


Untuk dapat withdrawal ke akun spesifik, dapat menambahkan fungsi berikut

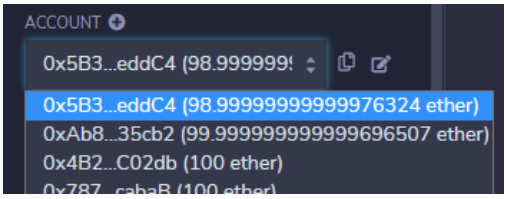
```
// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.1;

contract SendMoneyExample {
    uint public balanceReceived;

    function receiveMoney() public payable {
        balanceReceived += msg.value;
    }

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

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

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

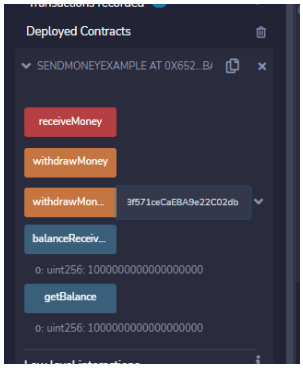
Redeploy smart contract dan kirim 1 ether, setelah itu pilih akun tujuan withdraw dan copy alamat akun tersebut



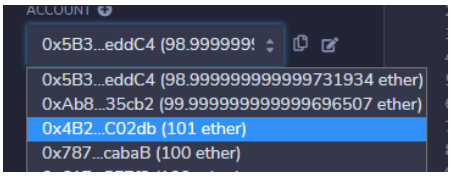


Kembali ke akun awal dan paste alamat tujuan di kolom withdrawMoneyTo dan klik boxnya





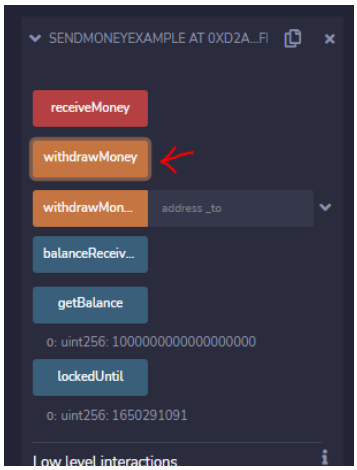
Akan terlihat akun tujuan telah mendapatkan ether withdrawal tersebut



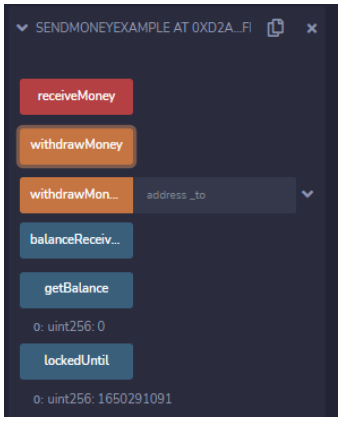
Untuk menambahkan penguncian withdrawal dapat dilakukan beberapa fungsi logika seperti

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.1;

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());
        }
    }
}
```

Redeploy smart contract seperti sebelumnya, dan apabila kita memilih fungsi withdraw balance akan tetap sama hingga satu menit



Setelah menunggu salama satu menit dan memilih fungsi withdraw lagi, withdraw baru dapat berjalan



II. LAB 2 : Shared Wallet

Langkah awal dengan membuat fungsi sederhana smart contract

```
//SPDX-License-Identifier: MIT

pragma solidity 0.8.1;

contract SharedWallet {

    function withdrawMoney(address payable _to, uint _amount) public {
        _to.transfer(_amount);
    }

        receive() external payable {
    }
}
```

Setelah itu dapat ditambahkan fungsi untuk membatasi hanya pemilik wallet saja yang dapat withdraw dari smart contract tersebut dengan menambahkan beberapa parameter di fungsi withdrawMoney() seperti onlyOwner

```
//SPDX-License-Identifier: MIT

pragma solidity 0.8.1;

contract SharedWallet {

    address owner;

    constructor() {
        owner = msg.sender;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "You are not allowed");
        _;
    }

    function withdrawMoney(address payable _to, uint _amount) public onlyOwner {
        _to.transfer(_amount);
    }

        receive() external payable {
    }
}
```

Namun mengatur logika owner secara manual dapat mempersulit pengaturan, maka kita dapat menggunakan fungsi smart contract owner yang sudah tersedia dari openZeppelin

```
//SPDX-License-Identifier: MIT

pragma solidity 0.8.1;

import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol";

contract SharedWallet is Ownable {

    function isOwner() internal view returns(bool) {
        return owner() == msg.sender;
    }

    function withdrawMoney(address payable _to, uint _amount) public onlyOwner {
        _to.transfer(_amount);
    }

    receive() external payable {
    }
}
```

Selanjutnya dimasukan mapping address yang tersimpan ke nomer spesifik agar kita tahu seberapa banyak seseorang dapat withdraw.

```
//SPDX-License-Identifier: MIT

pragma solidity 0.8.1;

import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol";

contract SharedWallet is Ownable {

    function isOwner() internal view returns(bool) {
        return owner() == msg.sender;
    }

    mapping(address => uint) public allowance;

    function addAllowance(address _who, uint _amount) public onlyOwner {
        allowance[_who] = _amount;
    }

    modifier ownerOrAllowed(uint _amount) {
        require(isOwner() || allowance[msg.sender] >= _amount, "You are not allowed!");
        _;
    }

    function withdrawMoney(address payable _to, uint _amount) public ownerOrAllowed(_amount) {
        require(_amount <= address(this).balance, "Contract doesn't own enough money");
        _to.transfer(_amount);
    }

    receive() external payable {
    }
}
```

Untuk menghindari withdrawal secara terus menerus, diperlukan pengurangan allowance untuk setiap orang selain owner.

```
    function reduceAllowance(address _who, uint _amount) internal ownerOrAllowed(_amount) {
        allowance[_who] -= _amount;
    }

    function withdrawMoney(address payable _to, uint _amount) public ownerOrAllowed(_amount) {
        require(_amount <= address(this).balance, "Contract doesn't own enough money");

        if(!isOwner()) {
            reduceAllowance(msg.sender, _amount);
        }

        _to.transfer(_amount);
    }
```

Setelah itu untuk mudah dimengerti, struktur smart contract tersebut dapat disusun sesuai contractnya yaitu Allowance dan SharedWallet.

```
pragma solidity 0.8.1;

import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol";

contract Allowance is Ownable {
    function isOwner() internal view returns(bool) {
        return owner() == msg.sender;
    }

    mapping(address => uint) public allowance;
    function setAllowance(address _who, uint _amount) public onlyOwner {
        allowance[_who] = _amount;
    }

    modifier ownerOrAllowed(uint _amount) {
        require(isOwner() || allowance[msg.sender] >= _amount, "You are not allowed!");
        _;
    }

    function reduceAllowance(address _who, uint _amount) internal ownerOrAllowed(_amount) {
        allowance[_who] -= _amount;
    }
}

contract SharedWallet is Allowance {
    function withdrawMoney(address payable _to, uint _amount) public ownerOrAllowed(_amount) {
        require(_amount <= address(this).balance, "Contract doesn't own enough money");

        if(!isOwner()) {
            reduceAllowance(msg.sender, _amount);
        }

        _to.transfer(_amount);
    }

    receive() external payable {
    }
```

Diperlukan event pada beberapa fungsi pada Allowance dan SharedWallet ketika seseorang deposit atau withdraw

```
event AllowanceChanged(address indexed _forWho, address indexed _byWhom, uint _oldAmount, uint _newAmount);
mapping(address => uint) public allowance;

function isOwner() internal view returns(bool) {
    return owner() == msg.sender;
}


function setAllowance(address _who, uint _amount) public onlyOwner {
    emit AllowanceChanged(_who, msg.sender, allowance[_who], _amount);
    allowance[_who] = _amount;
}

modifier ownerOrAllowed(uint _amount) {
    require(isOwner() || allowance[msg.sender] >= _amount, "You are not allowed!");
    _;
}

function reduceAllowance(address _who, uint _amount) internal ownerOrAllowed(_amount) {
    emit AllowanceChanged(_who, msg.sender, allowance[_who], allowance[_who] - _amount);
    allowance[_who] -= _amount;
}
```

```
contract SharedWallet is Allowance {

    event MoneySent(address indexed _beneficiary, uint _amount);
    event MoneyReceived(address indexed _from, uint _amount);

    function withdrawMoney(address payable _to, uint _amount) public ownerOrAllowed(_amount) {
        require(_amount <= address(this).balance, "Contract doesn't own enough money");

        if(!isOwner()) {
            reduceAllowance(msg.sender, _amount);
        }
        emit MoneySent(_to, _amount);
        _to.transfer(_amount);
    }

    receive() external payable {
        emit MoneyReceived(msg.sender, msg.value);
    }
}
```

Langkah selanjutnya kita memisah kedua smart contract tersebut menjadi dua file yang terpisah yaitu Allowance.sol

Home | Sharedwallet.sol | Allowance.sol

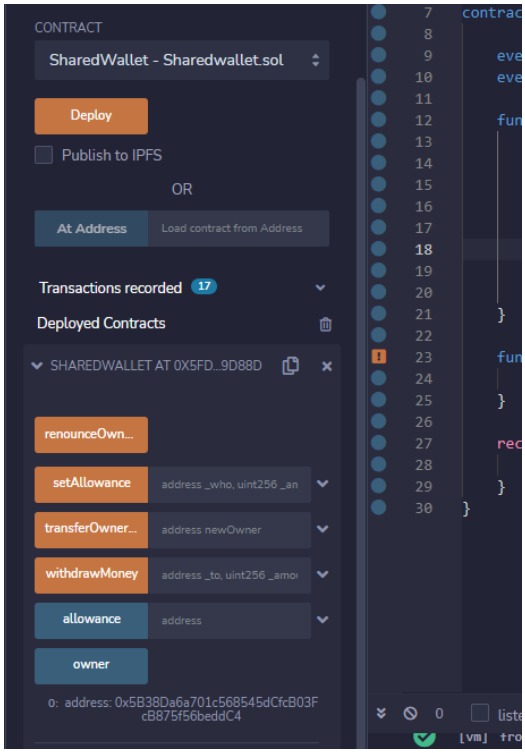
```
//SPDX-License-Identifier: MIT

pragma solidity 0.8.1;

import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol";

contract Allowance is Ownable {

    event AllowanceChanged(address indexed _forWho, address indexed _byWhom, uint _oldAmount, uint _newAmount);
    mapping(address => uint) public allowance;

    function isOwner() internal view returns(bool) {
        return owner() == msg.sender;
    }

    function setAllowance(address _who, uint _amount) public onlyOwner {
        emit AllowanceChanged(_who, msg.sender, allowance[_who], _amount);
        allowance[_who] = _amount;
    }

    modifier ownerOrAllowed(uint _amount) {
        require(isOwner() || allowance[msg.sender] >= _amount, "You are not allowed!");
        _;
    }

    function reduceAllowance(address _who, uint _amount) internal ownerOrAllowed(_amount) {
        emit AllowanceChanged(_who, msg.sender, allowance[_who], allowance[_who] - _amount);
        allowance[_who] -= _amount;
    }
}
```

Dan smart contract kedua yaitu Sharedwallet.sol dengan import fungsi dari Allowance.sol

Home | Sharedwallet.sol | Allowance.sol

```
//SPDX-License-Identifier: MIT

pragma solidity 0.8.1;

import "./Allowance.sol";

contract SharedWallet is Allowance {

    event MoneySent(address indexed _beneficiary, uint _amount);
    event MoneyReceived(address indexed _from, uint _amount);

    function withdrawMoney(address payable _to, uint _amount) public ownerOrAllowed(_amount) {
        require(_amount <= address(this).balance, "Contract doesn't own enough money");

        if(!isOwner()) {
            reduceAllowance(msg.sender, _amount);
        }

        emit MoneySent(_to, _amount);
        _to.transfer(_amount);
    }

    function renounceOwnership() public override onlyOwner {
        revert("can't renounceOwnership here"); //not possible with this smart contract
    }

    receive() external payable {
        emit MoneyReceived(msg.sender, msg.value);
    }
}
```

Saat akan mendeploy smart contract, pilih file Sharedwallet karena mencakup keseluruhan contract



III. LAB 3: Supply Chain

Untuk mencoba supply chain kita membutuhkan management smart contract untuk dapat menabahkan item bernama ItemManager.sol

```
pragma solidity ^0.6.0;

contract ItemManager{

    enum SupplyChainSteps{Created, Paid, Delivered}
    struct S_Item {
        ItemManager.SupplyChainSteps _step;
        string _identifier;
        uint _priceInWei;
    }

    mapping(uint => S_Item) public items;
    uint index;
    event SupplyChainStep(uint _itemIndex, uint _step);

    function createItem(string memory _identifier, uint _priceInWei) public {
        items[index]._priceInWei = _priceInWei;
        items[index]._step = SupplyChainSteps.Created;
        items[index]._identifier = _identifier;
        emit SupplyChainStep(index, uint(items[index]._step));
        index++;
    }

    function triggerPayment(uint _index) public payable {
        require(items[_index]._priceInWei <= msg.value, "Not fully paid");
        require(items[_index]._step == SupplyChainSteps.Created, "Item is further in the supply chain");
        items[_index]._step = SupplyChainSteps.Paid;
        emit SupplyChainStep(_index, uint(items[_index]._step));
    }

    function triggerDelivery(uint _index) public {
        require(items[_index]._step == SupplyChainSteps.Paid, "Item is further in the supply chain");
        items[_index]._step = SupplyChainSteps.Delivered;
        emit SupplyChainStep(_index, uint(items[_index]._step));
    }
}
```

Namun untuk mempermudah user untuk mengirim uang, kita dapat membuat sebuah smart contract baru untuk management bernama Item.sol

```
pragma solidity ^0.6.0;

import "./ItemManager.sol";

contract Item {

    uint public priceInWei;
    uint public paidWei;
    uint public index;
    ItemManager parentContract;

    constructor(ItemManager _parentContract, uint _priceInWei, uint _index) public {
        priceInWei = _priceInWei;
        index = _index;
        parentContract = _parentContract;
    }
    receive() external payable {
        require(msg.value == priceInWei, "We don't support partial payments");
        require(paidWei == 0, "Item is already paid!");
        paidWei += msg.value;
        (bool success, ) = address(parentContract).call{value:msg.value}(abi.encodeWithSignature("triggerPayment(uint256)", index));
        require(success, "Delivery did not work");
    }

    fallback () external {

    }
}
```

Dan merubah smart contract ItemManager untuk mengganti struct dengan mengimport smart contract Item sebelumnya. Dengan ini user dapat membayar langsung Item dari addres pada smart contract.

```
pragma solidity ^0.6.0;

import "./Item.sol";

contract ItemManager {
    struct S_Item {
        Item _item;
        ItemManager.SupplyChainSteps _step;
        string _identifier;
    }

    mapping(uint => S_Item) public items;
    uint index;
    enum SupplyChainSteps {Created, Paid, Delivered}
    event SupplyChainStep(uint _itemIndex, uint _step, address _address);

    function createItem(string memory _identifier, uint _priceInWei) public {
        Item item = new Item(this, _priceInWei, index);
        items[index]._item = item;
        items[index]._step = SupplyChainSteps.Created;
        items[index]._identifier = _identifier;
        emit SupplyChainStep(index, uint(items[index]._step), address(item));
        index++;
    }

    function triggerPayment(uint _index) public payable {
        Item item = items[_index]._item;
        require(address(item) == msg.sender, "Only items are allowed to update themselves");
        require(item.priceInWei() == msg.value, "Not fully paid yet");
        require(items[_index]._step == SupplyChainSteps.Created, "Item is further in the supply chain");
        items[_index]._step = SupplyChainSteps.Paid;
        emit SupplyChainStep(_index, uint(items[_index]._step), address(item));
    }

    function triggerDelivery(uint _index) public {
        require(items[_index]._step == SupplyChainSteps.Paid, "Item is further in the supply chain");
        items[_index]._step = SupplyChainSteps.Delivered;
        emit SupplyChainStep(_index, uint(items[_index]._step), address(items[_index]._item));
    }
}
```

Untuk mengamankan smart contract, dapat diterapkan fungsi owner yang mirip dengan OpenZeppelin dalam file Ownlable.sol

```
pragma solidity ^0.6.0;

contract Ownable {
    address public _owner;
    constructor () internal {
        _owner = msg.sender;
    }

    /**
    * @dev Throws if called by any account other than the owner.
    */
    modifier onlyOwner() {
        require(isOwner(), "Ownable: caller is not the owner");
        _;
    }

    /**
    * @dev Returns true if the caller is the current owner.
    */
    function isOwner() public view returns (bool) {
        return (msg.sender == _owner);
    }
}
```

Setelah itu, ubah setiap fungsi pada ItemManager yang berjalan hanya untuk owner.

```
Home | ItemManager.sol | Item.sol | Ownlable.sol

pragma solidity ^0.6.0;

import "./Ownable.sol";
import "./Item.sol";

contract ItemManager is Ownable{
    struct S_Item {
        Item _item;
        ItemManager.SupplyChainSteps _step;
        string _identifier;
    }

    mapping(uint => S_Item) public items;
    uint index;
    enum SupplyChainSteps {Created, Paid, Delivered}
    event SupplyChainStep(uint _itemIndex, uint _step, address _address);

    function createItem(string memory _identifier, uint _priceInWei) public onlyOwner {
        Item item = new Item(this, _priceInWei, index);
        items[index]._item = item;
        items[index]._step = SupplyChainSteps.Created;
        items[index]._identifier = _identifier;
        emit SupplyChainStep(index, uint(items[index]._step), address(item));
        index++;
    }

    function triggerPayment(uint _index) public payable {
        Item item = items[_index]._item;
        require(address(item) == msg.sender, "Only items are allowed to update themselves");
        require(item.priceInWei() == msg.value, "Not fully paid yet");
        require(items[_index]._step == SupplyChainSteps.Created, "Item is further in the supply chain");
        items[_index]._step = SupplyChainSteps.Paid;
        emit SupplyChainStep(_index, uint(items[_index]._step), address(item));
    }

    function triggerDelivery(uint _index) public onlyOwner {
        require(items[_index]._step == SupplyChainSteps.Paid, "Item is further in the supply chain");
        items[_index]._step = SupplyChainSteps.Delivered;
        emit SupplyChainStep(_index, uint(items[_index]._step), address(items[_index]._item));
    }
}
```

Membuat folder baru untuk penyimpanan data truffle

```
PS E:\Truffle> mkdir s06-eventtrigger


    Directory: E:\Truffle


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----         4/21/2022   5:08 PM                s06-eventtrigger


PS E:\Truffle> cd .\s06-eventtrigger\
PS E:\Truffle\s06-eventtrigger> ls
PS E:\Truffle\s06-eventtrigger>
```

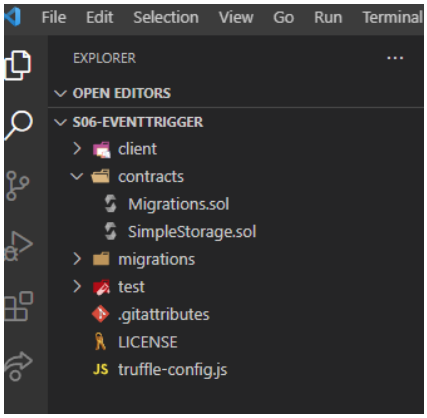
Langkah berikutnya yaitu menginstall Truffle di PowerShell

```
Select Administrator: C:\WINDOWS\system32\cmd.exe
PS E:\Truffle> cd .\s06-eventtrigger\
PS E:\Truffle\s06-eventtrigger> npm install -g truffle@5.1.8
npm WARN          mkdirp@0.5.1: Legacy versions of mkdirp are no longer supported. Please update to mkdirp 1.x. (Note
that the API surface has changed to use Promises in 1.x.)

changed 27 packages, and audited 28 packages in 7s

3 critical severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

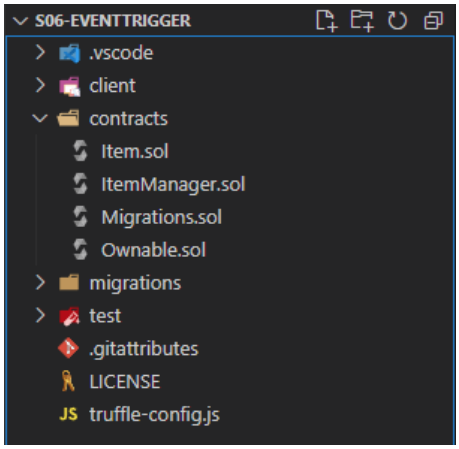
Setelah itu unbox react box pada folder tersebut, unbox ini juga dapat dilakukan di terminal VSCode menggunakan npx.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS E:\Truffle\s06-eventtrigger> npx truffle unbox react
√ Preparing to download box
√ Downloading
npm WARN old lockfile
npm WARN old lockfile The package-lock.json file was created with an old version of npm,
npm WARN old lockfile so supplemental metadata must be fetched from the registry.
npm WARN old lockfile
npm WARN old lockfile This is a one-time fix-up, please be patient...
npm WARN old lockfile
npm WARN deprecated @hapi/bourne@1.3.2: This version has been deprecated and is no longer
```

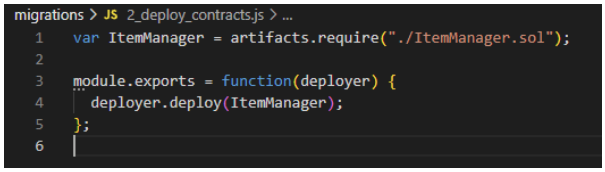
Maka otomatis file akan terdownload dan tersimpan di folder tersebut.



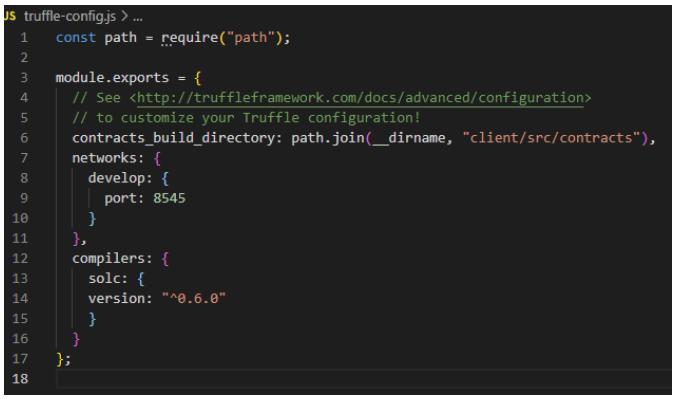
Setelah itu hapus SimpleStorage.sol, dan simpan file smart contract yang sudah kita buat sebelumnya.



File dalam folder migrations kita ubah menjadi



Versi compiler truffle juga perlu kita sesuaikan pada truffle-config.js



Setelah semuanya siap, maka jalankan develop console truffle dengan perintah : truffle develop

```
PS E:\Truffle\s06-eventtrigger> truffle develop
Truffle Develop started at http://127.0.0.1:8545/

Accounts:
(0) 0xd2afa94b7bac712e6cbfddbf0c15c2297d57f608
(1) 0xdb52a814c68bff0e6b9b7984906c1654d4f0641f
(2) 0x07c30d1b95ef9f4be1f011c697b80504b2c6058b
(3) 0x4647d471c9273d77608440a0e831538631319fc2
(4) 0x3219b9de3aa814f4a76ed10996cc4c568876407e
(5) 0x0b2b1dc34ca2507645896cce2240e7be7953975f
(6) 0x4ee58dc5876afbf50f967b93a394ca56827bd48d
(7) 0x5646c12ebfd02ac6fe731597a1c24f7d7646e599
(8) 0xc5d52fd4376afccc3e1e7d93737acd8213045568
(9) 0x71b1c37eab59a1a63a1a40652878752f94eb0665

Private Keys:
(0) ae8126f31dce762aa5897f5a5d682a4112dde45bfafe6e79331061f5ad664696
(1) 8ff1e208cebdbb8666f353e268f97834f6c0f417a736659c840c52befe4f885a
(2) 12620af58a5d73f698656dc7660c4a8bec8e0952e9d127f9f4add8967c2d8e51
(3) 676f6954470ce687e19634e703fecca7bdc4b5f54e6668ec6b0cf829d1e4bf6b
(4) 612e60cc32cca3028b38f94250e21b44d4f985c5478da6c2d6c2f4da2b0e1868
(5) 967c9828c7398d99b5c44c8fd40e0033b1fa2cfb50d3513849b96f56aeabce21
(6) f8eff13d2d6c860e824721eb88bc2b93f2f0997cca0ec2c00412d1dcd93be272
(7) f931f266df60eae81a4dcd6dcea561c0aca284d0ea9c736d1d48f24a6e8f1802
(8) 15be5868f129d5416d2286fab1e702c638ab3eff2a1029b699ccf31f30f2ca9c
(9) 012ac18c0e26fdfce7d0de6716185f321894eb8dee3898102c2ef47e82c7ed84

Mnemonic: divide wire anger buyer parrot sail siege pyramid claim deny pause valley

   Important   : This mnemonic was created for you by Truffle. It is not secure.
Ensure you do not use it on production blockchains, or else you risk losing funds.

truffle(develop)>
```

Dan menjalankan perintah migate.

```
truffle(develop)> migrate

Compiling your contracts...
===========================
⬇ Fetching solc version list from solc-bin. Attempt #1
⬇ Downloading compiler. Attempt #1.
> Compiling .\contracts\Item.sol
> Compiling .\contracts\ItemManager.sol
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\Ownable.sol
> Compiling .\contracts\Item.sol
> Compiling .\contracts\ItemManager.sol
> Compiling .\contracts\Ownable.sol
⬇ Fetching solc version list from solc-bin. Attempt #1
> Compilation warnings encountered:

    /E/Truffle/s06-eventtrigger/contracts/Item.sol: Warning: SPDX license identifier not provided in source file. Before
 publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SP
DX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.
,/E/Truffle/s06-eventtrigger/contracts/ItemManager.sol: Warning: SPDX license identifier not provided in source file. Be
fore publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use
 "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.
,/E/Truffle/s06-eventtrigger/contracts/Ownable.sol: Warning: SPDX license identifier not provided in source file. Before
 publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SP
DX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.

> Artifacts written to E:\Truffle\s06-eventtrigger\client\src\contracts
> Compiled successfully using:
   - solc: 0.6.12+commit.27d51765.Emscripten.clang


Starting migrations...
======================
> Network name:    'develop'
> Network id:      5777
> Block gas limit: 0x6691b7
```

Setelah itu ubah isi HTML agar dapat berinteraksi dengan smart contract di browser di file client/App.js

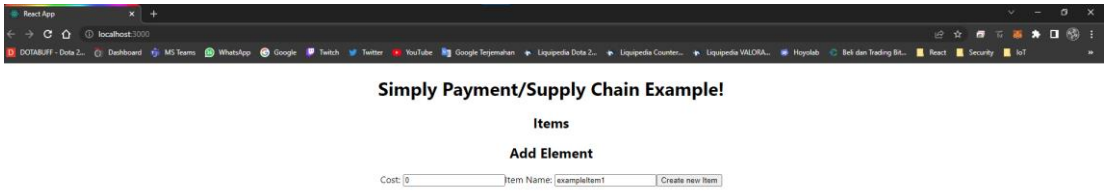
client > src > JS App.js > App > componentDidMount

```
import React, { Component } from "react";
import ItemManager from "./contracts/ItemManager.json";
import Item from "./contracts/Item.json";
import getWeb3 from "./getWeb3";

import "./App.css";

class App extends Component {
  state = {cost: 0, itemName: "exampleItem1", loaded:false};

  componentDidMount = async () => {
    try {
      // Get network provider and web3 instance.
      this.web3 = await getWeb3();

      // Use web3 to get the user's accounts.
      this.accounts = await this.web3.eth.getAccounts();

      // Get the contract instance.
      const networkId = await this.web3.eth.net.getId();

      this.itemManager = new this.web3.eth.Contract(
        ItemManager.abi,
        ItemManager.networks[networkId] && ItemManager.networks[networkId].address,
      );
      this.item = new this.web3.eth.Contract(
        Item.abi,
        Item.networks[networkId] && Item.networks[networkId].address,
      );
      this.setState({loaded:true});

```

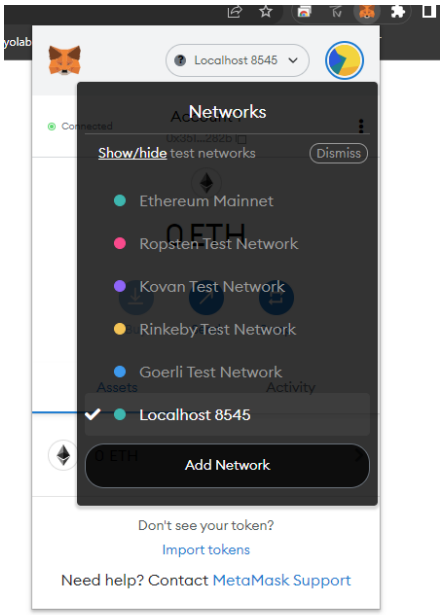
Mengubah isi render() dan menabahkan fungsi handleSubmit() dan handleInputChange()

```
client > src > JS App.js > App
handleSubmit = async () => {
  const { cost, itemName } = this.state;
  console.log(itemName, cost, this.itemManager);
  let result = await this.itemManager.methods.createItem(itemName, cost).send({ from: this.accounts[0] });
  console.log(result);
  alert("Send "+cost+" Wei to "+result.events.SupplyChainStep.returnValues._address);
};

handleInputChange = (event) => {
  const target = event.target;
  const value = target.type === 'checkbox' ? target.checked : target.value;
  const name = target.name;
  this.setState({
    [name]: value
  });
}

render() {
  if (!this.state.loaded) {
    return <div>Loading Web3, accounts, and contract...</div>;
  }
  return (
    <div className="App">
      <h1>Simply Payment/Supply Chain Example!</h1>
      <h2>Items</h2>
      <h2>Add Element</h2>
      Cost: <input type="text" name="cost" value={this.state.cost} onChange={this.handleInputChange} />
      Item Name: <input type="text" name="itemName" value={this.state.itemName} onChange={this.handleInputChange} />
      <button type="button" onClick={this.handleSubmit}>Create new Item</button>
    </div>
  );
}
```

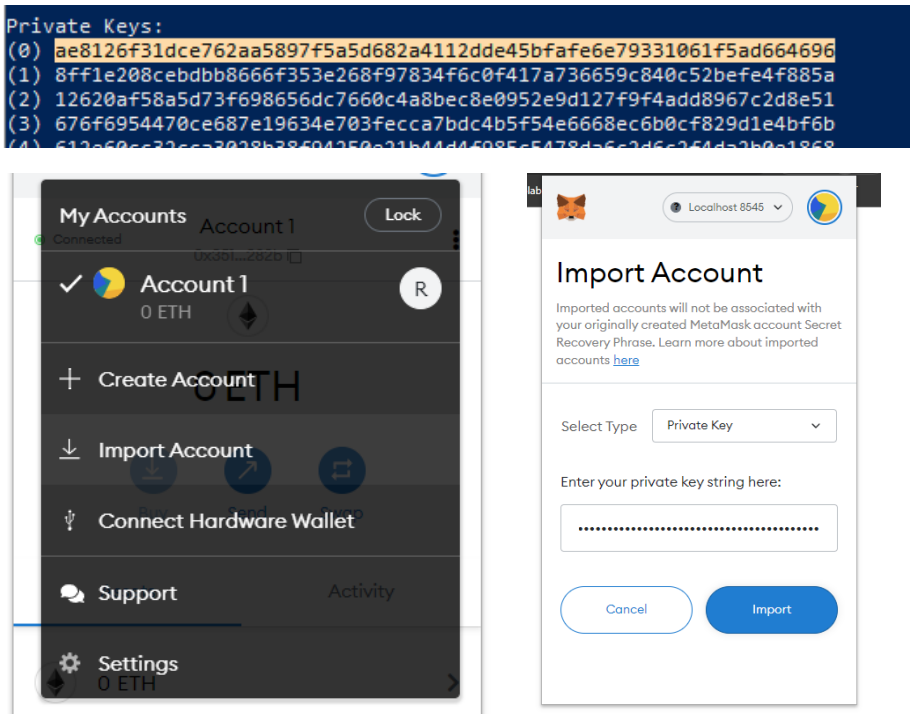
Setelah itu jalankan website di folder client dengan : npm start, dan seharusnya browser akan terbuka di port 3000



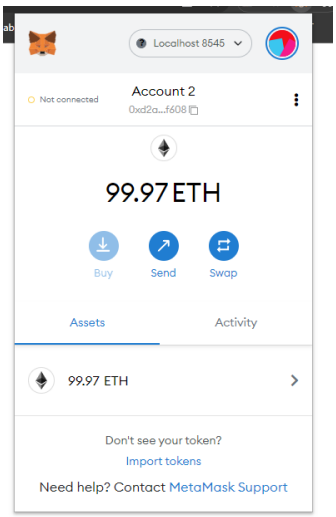
Selanjutnya adalah menghubungkan metamask dan aplikasi web ini, yaitu dengan memilih network yang sesuai



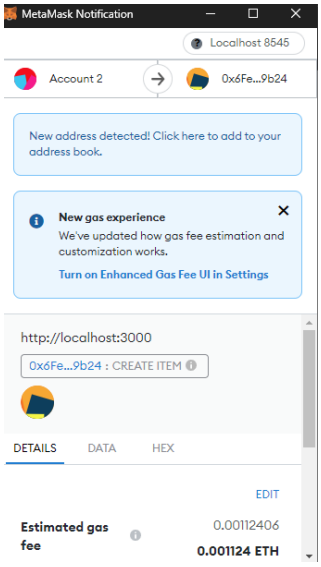
Berikutnya kita perlu memasukan private key dalam terminal developer ke metamask



Dan seharusnya akun akan berhasil masuk kedalam metamask



Berikutnya kita dapat menambahkan item ke smart contract pada kolom, dan setelah disubmit akan ada pop up



Untuk menambahkan listener pembayaran kita dapat menambah fungsi listenToPaymentEvent() pada App.js

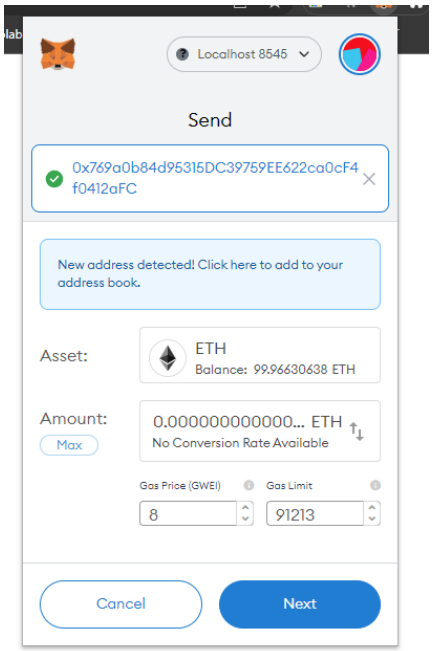
```
72  }
73
74  listenToPaymentEvent = () => {
75    let self = this;
76    this.itemManager.events.SupplyChainStep().on("data", async function(evt) {
77      if(evt.returnValues._step == 1) {
78        let item = await self.itemManager.methods.items(evt.returnValues._itemIndex).call();
79        console.log(item);
80        alert("Item " + item._identifier + " was paid, deliver it now!");
81      };
82      console.log(evt);
83    });
84  }
85
```

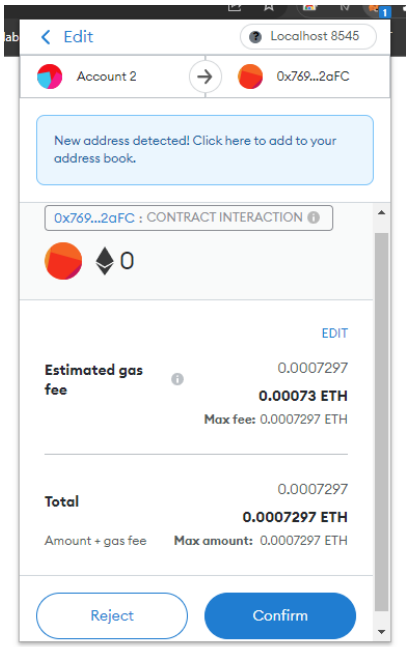
Dan memanggilnya di componentDidMount()

```
24      );
25
26      this.Item = new this.web3.eth.Contract(
27        Item.abi,
28        Item.networks[this.networkId] && Item.networks[this.networkId].address,
29      );
30
31      this.listenToPaymentEvent();
32      this.setState({loaded:true});
33
```

Dan bila seseorang mengirim ether maka muncul popup untuk mengirim pembayaran. Contoh membuat item dengan 1000 wei

Seseorang perlu mengirim 1000 wei ke 0x769a0b84d95315DC39759EE622ca0cF4f0412aFC





Setelah di konfirmasi maka akan muncul pop up bahwa item telah terbayar.