Presale ERC20 Token Contract

Di X sempat rame token presale dengan kirim ETH ke contract. Guide ini akan menjelaskan step-by-step cara bikinnya.

Contents

Prerequisites

- Pemahaman tentang ERC20
- Pemahaman tentang Eth transfer
- Pemahaman tentang data type Solidity
- Pemahaman tentang Hardhat commands (optional, bisa via Remix online compiler)

Concept

Contract presale yang dibuka dedicated untuk 1 token. Harga token dalam Eth harus sudah ditentukan di contract agar setiap buyer bisa diketahui alokasi tokennya. Buyer bisa langsung menerima token saat dibeli atau setelah presale selesai, token langsung dibagikan ke buyer. Kapan buyer menerima token, menentukan kapan Token harus ada di dalam contract.

Optional modification:

- Menentukan token supply maksimum
- Membuat beberapa tahap presale dengan harga berbeda
- Membuat whitelist

Feel free to try it yourself before reading the guide.

Contract Guide

Setelah bikin project Hardhat baru, bikin 2 contract: Token (untuk testing, simple ERC20) dan Presale. Guide ini akan fokus ke presale contract.

1. Basic contract structure dan import library.
    a. IERC20: Interface ERC20 untuk panggil basic function dari token address.
    b. Ownable: Untuk **onlyOwner** modifier (access control).
    c. ReentrancyGuard: Untuk security, jaga-jaga biar ngga ada celah transfer token lebih dari sekali.
    d. SafeMath: Untuk kalkulasi uint yang aman di contract.

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.19;

import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/security/ReentrancyGuard.sol";
import "@openzeppelin/contracts/utils/math/SafeMath.sol";

contract Presale is Ownable, ReentrancyGuard {
}
```

2. Constructor. Set token address, token supply allocated for presale, token price in Eth (wei). Pakai library SafeMath untuk uint256.

```
contract Presale is Ownable, ReentrancyGuard {
    using SafeMath for uint256;

    IERC20 token; // token address
    uint256 public presaleSupply; // amount of token left available for presale
    uint256 public presalePrice; // amount of wei (Eth) for 1 token

    constructor(address _token, uint256 _presaleSupply, uint256 _presalePrice) {
        token = IERC20(_token);
```

```
        presaleSupply = _presaleSupply;
        presalePrice = _presalePrice;
    }
}
```

3. Bikin function untuk start dan end presale. Nilainya disimpan di state variable.

```
    bool isActive;
    function startPresale() external onlyOwner {
        token.transferFrom(msg.sender, address(this), presaleSupply);
        isActive = true;
    }

    function endPresale() external onlyOwner {
        isActive = false;
    }
```

4. Bikin function receive untuk handle Eth yang dikirim ke contract. Cek apakah presale sudah active dan sisa token mencukupi.

   Di sini rumus untuk tahu jumlah token yang dibeli adalah jumlah Eth dikali decimal token per harga token karena harga yang di set di **presalePrice** itu harga per 1 token **bukan dalam satuan desimal-nya**. Kalau rumusnya dibagi baru dikali, **ngga akan bisa beli fractional token**.

   Untuk bisa kasih error message dengan variable, kita bisa bikin custom **error**. Di sini tujuannya biar buyer tau sisa supply token yang dapat dibeli. Lalu bikin event untuk di emit setiap ada yang beli.

```
    error InsufficientSupply(uint256 _presaleSupplyLeft);
    event Purchase(address _buyer, uint256 _tokenAmt);

    // to receive Eth
    receive() external payable nonReentrant {
        require(isActive == true, "Inactive presale");
        uint256 boughtAmt = msg.value.mul(10 ** 18).div(presalePrice);

        if (boughtAmt > presaleSupply) {
            revert InsufficientSupply({_presaleSupplyLeft: presaleSupply});
        }

        token.transfer(msg.sender, boughtAmt);
        presaleSupply = presaleSupply.sub(boughtAmt);

        emit Purchase(msg.sender, boughtAmt);
    }
```

5. Bikin withdraw function untuk owner.

```
    function withdrawAllEth() external onlyOwner {
        (bool sent, ) = msg.sender.call{value: address(this).balance}("");
        require(sent, "Failed to withdraw Ether");
    }

    function withdrawAllToken() external onlyOwner {
        token.transfer(msg.sender, token.balanceOf(address(this)));
    }
```

6. Run testing. Code lengkap bisa cek di Github.