Lesson 9

Hardhat smart contract lottery

Hikmah Nisya - 1103184094
Radzis Araaf Jaya Jamaludin - 1103184234
Raudhatul Rafiqah Assyahiddini - 1103180225

Apa itu Hardhat?

Hardhat adalah Environment yang digunakan pengembang untuk mengkompilasi, menguji, menyebarkan, dan men-debug dApps berbasis Ethereum. Ini berarti membantu pengembang dan pembuat kode untuk mengelola banyak tugas asli dalam mengembangkan kontrak pintar.

Hardhat hadir dengan jaringan Ethereum lokal yang sudah dibangun sebelumnya dengan fokus pada pengembangan. Jaringan ditentukan untuk debugging Soliditas dan fitur pesan kesalahan, jejak tumpukan, antara lain. Dengan demikian, lingkungan ini sangat membantu dalam memungkinkan pengembang untuk memahami di mana dApps mereka gagal dan bagaimana mereka dapat memecahkan masalah.

Langkah Verify smart contract menggunakan hardhat

1. Set up Hardhat
2. Create NFT smart contract
3. Create a deployment script
4. Modify hardhat.config.js
5. Run script

Set up

- Install hardhat menggunakan yarn atau npm

```
PS D:\Kuliah\Semester 8\Blockchain\hardhat-smartcontract-lottery-fcc-main\hardhat-smartcontrac
t-lottery-fcc-main> yarn add --dev hardhat
```

Selanjutnya kita lakukan deploy hardhat-deploy untuk penerapan serta include plugin tautan dana untuk kontrak dengan link.

```
PS D:\Kuliah\Semester 8\Blockchain\hardhat-smartcontract-lottery-fcc-main\hardhat-smartcontract-lottery-fcc-main> yarn add hardhat-deploy @appliedblockchain/chainli
nk-plugins-fund-link --dev
>>
```

Hardhat.config.js

JS hardhat.config.js > ...

```
1   require("@nomiclabs/hardhat-waffle")
2   require("@nomiclabs/hardhat-etherscan")
3   require("hardhat-deploy")
4   require("solidity-coverage")
5   require("hardhat-gas-reporter")
6   require("hardhat-contract-sizer")
7   require("dotenv").config()
8
```

Build sebuah blockchain dengan based decentralized lottery

SET UP

```
pragma solidity ^0.8.7;
import "guthub.com smartcontractkit/chainlink/ecm-contracts/src/v0/ChainlinkClient.sol"

contract Lottery is ChainingClient {
    enum Lottery_state { OPEN, CLOSE, CALCULATING_WINNER}
    LOTTERY_STATE public lottery_state;
    address payable[] public players;
    uint256 public lotteryid
}
```

Rafell. sol

```
1
2
3   // SPDX-License-Identifier: MIT
4
5   pragma solidity ^0.8.7;
6
7   import "@chainlink/contracts/src/v0.8/interfaces/VRFCoordinatorV2Interface.sol";
8   import "@chainlink/contracts/src/v0.8/VRFConsumerBaseV2.sol";
9   import "@chainlink/contracts/src/v0.8/interfaces/KeeperCompatibleInterface.sol";
10  import "hardhat/console.sol";
11
12  error Raffle__UpkeepNotNeeded(uint256 currentBalance, uint256 numPlayers, uint256 raf
13  error Raffle__TransferFailed();
14  error Raffle__SendMoreToEnterRaffle();
15  error Raffle__RaffleNotOpen();
16
17  /**@title A sample Raffle Contract
18   * @author Patrick Collins
19   * @notice This contract is for creating a sample raffle contract
20   * @dev This implements the Chainlink VRF Version 2
21   */
22  contract Raffle is VRFConsumerBaseV2, KeeperCompatibleInterface {
23      /* Type declarations */
24      enum RaffleState {
25          OPEN,
26          CALCULATING
27      }
28      /* State variables */
29      // Chainlink VRF Variables
```

Set chainlink vrf variable

```
/* State variables */
// Chainlink VRF Variables
VRFCoordinatorV2Interface private immutable i_vrfCoordinator;
uint64 private immutable i_subscriptionId;
bytes32 private immutable i_gasLane;
uint32 private immutable i_callbackGasLimit;
uint16 private constant REQUEST_CONFIRMATIONS = 3;
uint32 private constant NUM_WORDS = 1;

// Lottery Variables
uint256 private immutable i_interval;
uint256 private s_lastTimeStamp;
address private s_recentWinner;
uint256 private i_entranceFee;
address payable[] private s_players;
RaffleState private s_raffleState;

/* Events */
event RequestedRaffleWinner(uint256 indexed requestId);
event RaffleEnter(address indexed player);
event WinnerPicked(address indexed player);
```

Set masukan raffle

```
function enterRaffle() public payable {
    // require(msg.value >= i_entranceFee, "Not enough value sent");
    // require(s_raffleState == RaffleState.OPEN, "Raffle is not open");
    if (msg.value < i_entranceFee) {
        revert Raffle__SendMoreToEnterRaffle();
    }
    if (s_raffleState != RaffleState.OPEN) {
        revert Raffle__RaffleNotOpen();
    }
    s_players.push(payable(msg.sender));
    // Emit an event when we update a dynamic array or mapping
    // Named events with the function name reversed
    emit RaffleEnter(msg.sender);
}
```

```
function fulfillRandomWords(
    uint256, /* requestId */
    uint256[] memory randomWords
) internal override {
    // s_players size 10
    // randomNumber 202
    // 202 % 10 ? what's doesn't divide evenly into 202?
    // 20 * 10 = 200
    // 2
    // 202 % 10 = 2
    uint256 indexOfWinner = randomWords[0] % s_players.length;
    address payable recentWinner = s_players[indexOfWinner];
    s_recentWinner = recentWinner;
    s_players = new address payable[](0);
    s_raffleState = RaffleState.OPEN;
    s_lastTimeStamp = block.timestamp;
    (bool success, ) = recentWinner.call{value: address(this).balance}("");
    // require(success, "Transfer failed");
    if (!success) {
        revert Raffle__TransferFailed();
    }
    emit WinnerPicked(recentWinner);
}
```

Set pick winner