BÀI THỰC HÀNH SỐ 5

Chủ đề: Blockchain

Yêu cầu: *Nộp báo cáo bao gồm Code + file Doc ghi hình ảnh kết quả các bài tập (ảnh chụp màn hình kết quả chạy chương trình)*

Bài số 1: Kiến thức cơ sở vê Blockchain và Ethereum

- **Blockchain là gì?** Một sổ cái phân tán, không thể thay đổi, lưu trữ dữ liệu dưới dạng khối liên kết. Ethereum là blockchain hỗ trợ smart contract (mã tự động thực thi khi điều kiện đạt).
- **Smart contract:** Như một chương trình chạy trên blockchain, không cần trung gian.
- **dApp:** Ứng dụng sử dụng smart contract, thường có frontend (web) kết nối với blockchain.
- Kiến thức blockchain, ngôn ngữ lập trình Solidity, Node.js, Script và môi trường Remix/VS code.
- Các công cụ nguồn mở: Ganace, Truffle, Metamask, Ether.js/Web3.js

Bài ví dụ: Tao một DApp đơn giản: **Simple Storage** (Lưu trữ đơn giản), cho phép bạn đọc và ghi một số nguyên lên blockchain.

<u>Bước 1:</u> Cài đặt môi trường phát triển

Đảm bảo bạn đã cài đặt **Node.js, npm, Truffle** và **Ganache CLI** toàn cục.

1. **Cài Node.js và npm:**
    - Tải từ nodejs.org (phiên bản LTS, hiện tại ~20.x).
    - Kiểm tra: Mở terminal (Command Prompt trên Windows *node -v* và *npm -v*.
2. **Cài Truffle:** Framework để phát triển, compile, deploy smart contract.
    - Chạy: *npm install -g truffle*.
    - Kiểm tra*: truffle version*.

3. **Cài Ganache:** Blockchain local để test (như một Ethereum giả lập trên máy cục bộ).
    - Tải từ **trufflesuite.com/ganache** (phiên bản desktop).
    - Chạy ứng dụng, nó sẽ tạo 10 tài khoản test với ETH giả.

Bước 2: Khởi tạo Dự án Truffle

Bash

Tạo và chuyển vào thư mục dự án

mkdir simple-storage-dapp

cd simple-storage-dapp

Khởi tạo dự án Truffle

truffle init

Bước 3: Smart Contract (Solidity)

3.1. Viết Contract

Tạo tệp SimpleStorage.sol trong thư mục contracts/ và dán mã Solidity sau:

```
// contracts/SimpleStorage.sol
// Chỉ định giấy phép (khắc phục cảnh báo SPDX)
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract SimpleStorage {
    uint256 private data;
    // Hàm set: Ghi dữ liệu lên blockchain (Transaction)
    function set(uint256 newData) public {
        data = newData;
    }
    // Hàm get: Đọc dữ liệu từ blockchain (Call)
    function get() public view returns (uint256) {
        return data;
```

```
    }
````}
```

### 3.2. Viết Migration (Di trú)

Tạo tệp *2_deploy_simplestorage.js* trong thư mục migrations/ và dán mã sau:

JavaScript

```
// migrations/2_deploy_simplestorage.js
const SimpleStorage = artifacts.require("SimpleStorage");
module.exports = function (deployer) {
  // Triển khai contract SimpleStorage
  deployer.deploy(SimpleStorage);
};
```

3.3. Biên dịch và Cấu hình Truffle

1. **Chỉnh sửa truffle-config.js:** Cấu hình mạng development và trình biên dịch.

   JavaScript

   ```
   // truffle-config.js
   module.exports = {
    networks: {
      development: {
        host: "127.0.0.1",
        port: 8545,
        network_id: 5777, // Sẽ sử dụng 5777 làm Chain ID mặc định của Ganache
      },
    },
    compilers: {
      solc: {
        version: "0.8.0", // Phiên bản phải khớp với pragma trong contract
   ```
   ```

```
    }
  }
};
```

2. **Biên dịch Contract:**

truffle compile

3.4. Triển khai hợp đồng thông minh

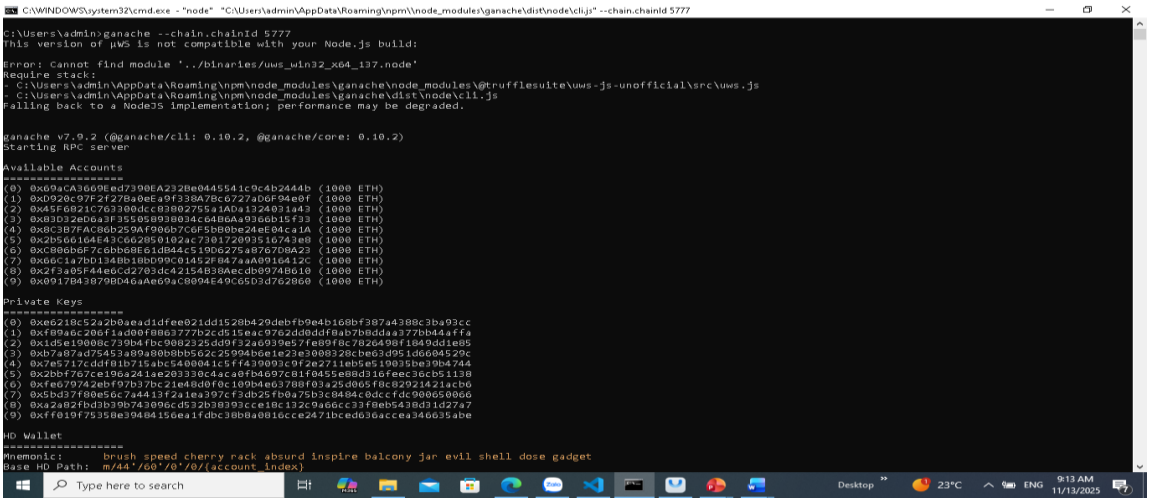
- Khởi động Ganache trên 1 cửa sổ riêng.

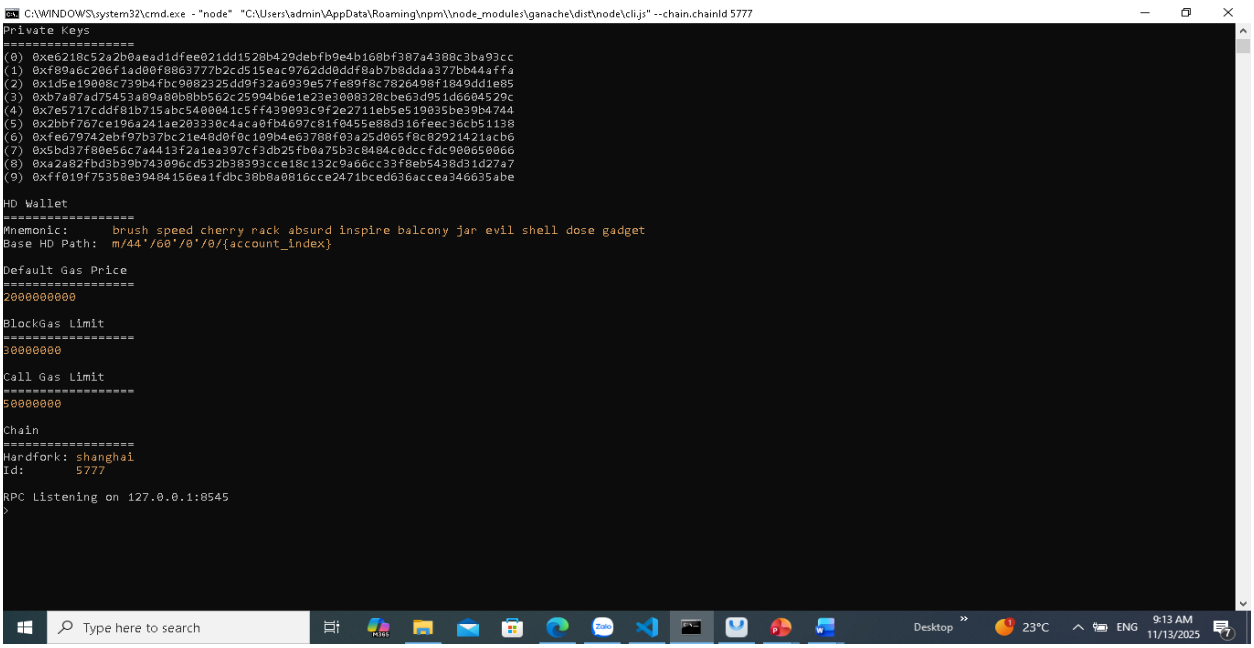
Chạy Ganache với Chain ID cố định 5777 để khớp với cấu hình Truffle.

Bash

ganache --chain.chainId 5777

Kết quả:





Vậy Ganache đã được khởi động, đang nghe kết nối và sẵn sáng làm việc.

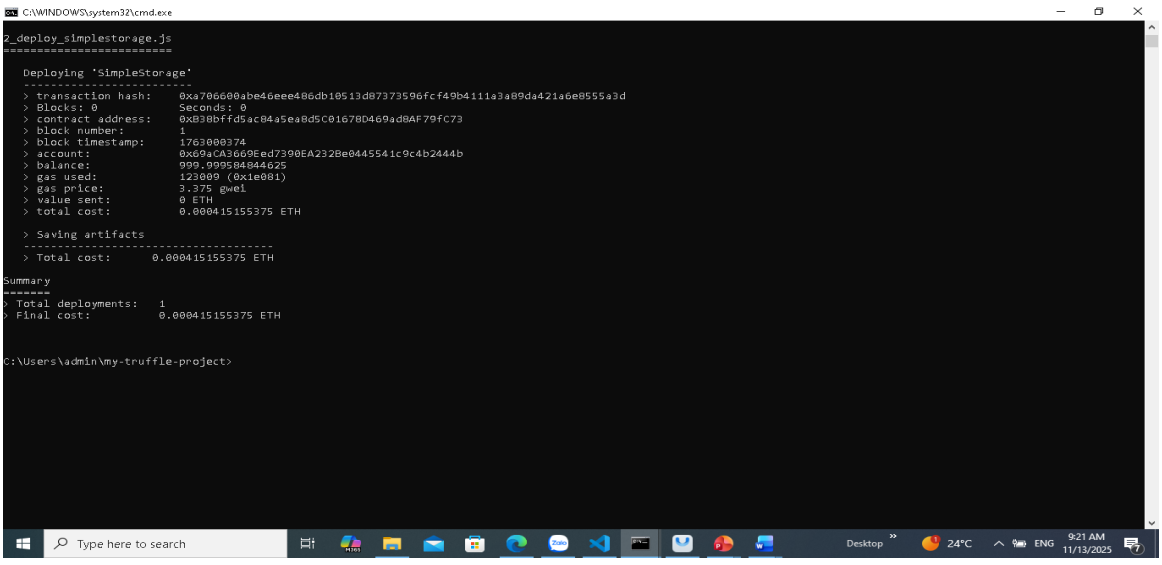
- **Triển khai Contract**

Trên cửa số khác chạy lệnh di trú để triển khai hợp đồng lên Ganache. **Ghi lại contract address**.

truffle migrate --network development

Lưu ý: Phải đang ffuwngs trong thư mục dự án **(my-truffle-project)**.

Kết quả:



- **Truy cập Hợp đồng đã triển khai**

Trong console, lấy một thể hiện (instance) của hợp đồng đã được triển khai:

1. **Lấy Contract Abstraction:** Tải thông tin về Smart Contract của bạn.

   JavaScript

   let SimpleStorage = artifacts.require("SimpleStorage");
2. **Lấy Instance đã triển khai:** Sử dụng hàm deployed() để lấy thể hiện (instance) của hợp đồng đã triển khai trên mạng development.

   JavaScript

   let instance = await SimpleStorage.deployed();

- **Tương tác với Hợp đồng**

Gọi các hàm get và set trên biến instance.

+ Đọc giá trị ban đầu (get)

Khi hợp đồng được triển khai, biến data (kiểu uint256) mặc định sẽ là **0**.

JavaScript

let currentValue = await instance.get();

console.log(currentValue.toString());

- **Kết quả mong đợi:** 0

+ Cập nhật giá trị (set)

Hàm set là một giao dịch (transaction) vì nó thay đổi trạng thái của blockchain, do đó, nó sẽ tốn Gas.

JavaScript

await instance.set(99);

- **Kết quả:** Sau khi chạy lệnh, sẽ có một đối tượng **Transaction Receipt** (biên lai giao dịch) phức tạp, xác nhận rằng giao dịch đã được Ganache xử lý.

+ Xác minh giá trị mới (get)

Gọi lại hàm get để xác nhận rằng giá trị đã được cập nhật:

JavaScript

let newValue = await instance.get();

console.log(newValue.toString());

- **Kết quả mong đợi:** 99

4. Thoát Console

Sau khi hoàn thành, bạn có thể thoát khỏi Truffle Console bằng lệnh:

JavaScript

.exit

Hoặc đơn giản là nhấn **Ctrl + C** hai lần.

NHư vậy hợp đồng đã được test thành công. Tiếp đến sẽ tạo Front end thông qua sử dụng ví điện tử Metamask.

<u>Bước 4:</u> Cài đặt và cấu hình MetaMask

- Cài đặt Metamask:

 Truy cập Trang Tải xuống Chính thức: Mở trình duyệt (Chrome, Firefox, v.v.) và truy cập https://metamask.io/download/.

 Chọn Trình duyệt: Nhấp vào "Install MetaMask for [Tên Trình duyệt]".

 Thêm vào Trình duyệt: Bạn sẽ được chuyển đến cửa hàng tiện ích. Nhấp vào "Thêm vào [Trình duyệt]" và xác nhận "Thêm tiện ích".

- Thiết lập Ví

Sau khi cài đặt, MetaMask sẽ mở ra. Nhấp vào **"Get Started"** (Bắt đầu).

1. **Đồng ý Điều khoản:** Nhấp vào **"I Agree"** (Tôi đồng ý).
2. **Tạo Ví:** Chọn **"Create a new wallet"** (Tạo Ví mới).
3. **Tạo Mật khẩu:** Tạo một mật khẩu mạnh. Mật khẩu này dùng để mở khóa ứng dụng MetaMask trên thiết bị của bạn.
4. **Ghi lại Cụm từ Khôi phục Bí mật (SRP): Đây là bước quan trọng nhất.**
    - Nhấp vào để xem 12 từ bí mật.
    - **Viết tay** cụm từ này ra giấy và cất giữ ở nơi an toàn. **Không** lưu trữ kỹ thuật số.

5. **Xác nhận SRP:** Sắp xếp các từ theo đúng thứ tự để xác nhận.
6. **Hoàn tất:** Nhấn **"All Done"** (Hoàn tất).

MetaMask đã sẵn sàng và mặc định kết nối với **Ethereum Mainnet**.

<u>Bước 5:</u> Cấu hình Mạng Ganache Cục bộ

Để DApp hoạt động, người sử dụng cần kết nối MetaMask với mạng Ganache cục bộ đã được triển khai hợp đồng.

- **Chuẩn bị Thông tin Mạng Ganache**

Giả sử bạn đã khởi động Ganache với Chain ID cố định là 5777 (hoặc kiểm tra terminal Ganache của bạn để biết Chain ID chính xác):

- **Network Name:** Tên tùy chọn (ví dụ: Ganache Local 5777)
- **New RPC URL:** http://127.0.0.1:8545 (Địa chỉ cục bộ mặc định)
- **Chain ID:** 5777 (Giá trị Ganache đang sử dụng)
- **Currency Symbol:** ETH

- **Thêm Mạng vào MetaMask**

1. **Mở MetaMask:** Nhấp vào biểu tượng con cáo.
2. **Mở Menu Mạng:** Nhấp vào menu thả xuống tên mạng ở góc trên cùng (thường là "Ethereum Mainnet").
3. **Thêm Mạng mới:** Chọn **"Add network"** (Thêm mạng).
4. **Chọn Thêm Thủ công:** Chọn **"Add a network manually"** (Thêm mạng thủ công).
5. **Nhập Thông tin Cấu hình:** Điền các thông tin đã chuẩn bị ở Bước 1.

Trường	**Giá trị (Ví dụ)**
Network name	Ganache Local 5777
New RPC URL	http://127.0.0.1:8545
Chain ID	5777

Trường	Giá trị (Ví dụ)
Currency symbol (optional)	ETH
Block explorer URL (optional)	(Để trống)

Lưu ý: ***Phải tra cứu lại Chain ID (trên lệnh chạy Ganache hoặc thoogn tin sau khi chạy lệnh ganache)***

Bước 6: Xây dựng Front-end (Ethers.js)

- **Cấu trúc Front-end**

Tạo thư mục client/ trong thư mục gốc của dự án và thêm hai tệp:

mkdir client

touch client/index.html client/script.js

Nên dùng notepad tạo tệp thay lệnh *touch* để tạo 2 tệp.

- **Mã HTML (client/index.html)**

Dán mã giao diện cơ bản sau:

HTML

```
<!DOCTYPE html>
<html lang="vi">
<head>
   <meta charset="UTF-8">
   <title>Simple Storage DApp</title>
   <script src="https://cdn.ethers.io/lib/ethers-5.7.0.umd.min.js" type="application/javascript"></script>
   <script src="script.js" defer></script>
</head>
<body>
   <h1>Simple Storage DApp (Truffle + Ethers.js)</h1>
   <button id="connectWallet">1. Kết nối Ví (MetaMask)</button>
```

```
    <p>Trạng thái Ví: <span id="walletStatus">Chưa kết nối</span></p>
    <hr>
    <h2>Đọc/Ghi Dữ liệu</h2>
    <button id="readData">Lấy Giá trị Hiện tại</button>
    <p>Giá trị Đã lưu: <span id="currentValue">...</span></p>
    <br>
    <input type="number" id="newValue" placeholder="Nhập số mới">
    <button id="writeData">Lưu Giá trị Mới</button>
    <p id="transactionStatus"></p>
</body>
</html>
```

- **Mã JavaScript (client/script.js)**

Sao chép contract address và ABI (từ tệp build/contracts/SimpleStorage.json) vào phần khai báo.

JavaScript

```
// client/script.js
//  THAY THẾ bằng địa chỉ hợp đồng của bạn từ truffle migrate
const CONTRACT_ADDRESS =
            "0xe78A0F7E598Cc8b0Bb87894B0F60dD2a88d6a8Ab";
//  Dán mảng ABI JSON từ SimpleStorage.json
const CONTRACT_ABI = [
    {
        "inputs": [],
        "name": "get",
        "outputs": [{"internalType": "uint256","name": "","type": "uint256"}],
        "stateMutability": "view",
        "type": "function"
```

```
    },
    {
        "inputs": [{"internalType": "uint256","name": "newData","type": "uint256"}],
        "name": "set",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    }
];
let signer = null;
let simpleStorageContract = null;
// --- 1. KẾT NỐI VÍ ---
document.getElementById('connectWallet').addEventListener('click',
connectWallet);
async function connectWallet() {
    if (typeof window.ethereum === 'undefined') {
        alert("Vui lòng cài đặt MetaMask!");
        return;
    }
        try {
        // Yêu cầu kết nối và lấy các tài khoản
        const provider = new ethers.providers.Web3Provider(window.ethereum);
        await provider.send("eth_requestAccounts", []);
              signer = provider.getSigner();
        // Khởi tạo Contract Instance (với signer để gửi giao dịch)
        simpleStorageContract = new ethers.Contract(CONTRACT_ADDRESS,
CONTRACT_ABI, signer);
```

```
        document.getElementById('walletStatus').textContent = `Đã kết nối: ${await signer.getAddress()}`;
        alert("Kết nối ví thành công!");
    } catch (error) {
        console.error("Lỗi kết nối:", error);
        alert("Lỗi kết nối ví. Kiểm tra MetaMask và mạng Ganache.");
    }
}
// --- 2. ĐỌC DỮ LIỆU ---
document.getElementById('readData').addEventListener('click', readData);
async function readData() {
    if (!simpleStorageContract) { return alert("Kết nối ví trước!"); }
      try {
        const data = await simpleStorageContract.get();
        document.getElementById('currentValue').textContent = data.toString();
    } catch (error) {
        console.error("Lỗi khi đọc dữ liệu:", error);
        document.getElementById('currentValue').textContent = "Lỗi!";
    }
}
// --- 3. GHI DỮ LIỆU ---
document.getElementById('writeData').addEventListener('click', writeData);
async function writeData() {
    if (!simpleStorageContract) { return alert("Kết nối ví trước!"); }

    const newValue = document.getElementById('newValue').value;
    if (!newValue || isNaN(newValue)) { return alert("Nhập số hợp lệ."); }
```

```
    const statusElement = document.getElementById('transactionStatus');
    statusElement.textContent = "Đang gửi giao dịch... Vui lòng xác nhận trong MetaMask.";
    try {
        // Gửi giao dịch
        const tx = await simpleStorageContract.set(newValue);
        statusElement.textContent = `Giao dịch đã gửi: ${tx.hash}. Đang chờ xác nhận...`;
        // Chờ giao dịch được xác nhận (đưa vào block)
        await tx.wait();
        statusElement.textContent = `Giao dịch HOÀN TẤT! Giá trị mới: ${newValue}`;
        readData(); // Cập nhật giao diện
    } catch (error) {
        console.error("Lỗi giao dịch:", error);
        statusElement.textContent = `Giao dịch THẤT BẠI: ${error.message}`;
    }
}
```

- **Chạy Ứng dụng**

1. **Chạy Ganache:** Đảm bảo Ganache (ganache --chain.chainId 5777) vẫn chạy trong một terminal.
2. **Phục vụ Front-end:** Sử dụng Live Server trong VS Code (hoặc python -m http.server) để mở client/index.html.
3. **Tương tác:**
    - Nhấp vào **"Kết nối Ví (MetaMask)"**.
    - Nhấp vào **"Lấy Giá trị Hiện tại"**.
    - Nhập số và nhấp vào **"Lưu Giá trị Mới"** (xác nhận trong MetaMask).

Bài 2: Thực hiện Bài 1 trong môi trường VS Code

Bước 1. Thiết lập Môi trường Phát triển

Đảm bảo bạn đã cài đặt các công cụ sau: **Node.js, npm, VS Code.**

1.1. Cài đặt Công cụ Toàn cục

Mở Terminal tích hợp trong VS Code (Ctrl + `) và cài đặt Truffle và Ganache CLI:

```
npm install -g truffle
npm install -g ganache
```

1.2. Khởi tạo Dự án Truffle

Tạo thư mục dự án và khởi tạo Truffle:

```
# Tạo thư mục dự án
mkdir simple-storage-dapp-full
cd simple-storage-dapp-full

# Khởi tạo Truffle
truffle init
```

2. Phát triển Back-end (Solidity & Truffle)

2.1. Viết Smart Contract (SimpleStorage.sol)

Tạo tệp **contracts/SimpleStorage.sol** và dán mã sau:

```
// contracts/SimpleStorage.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract SimpleStorage {
    uint256 private data;
    function set(uint256 newData) public {
        data = newData;
    }
    function get() public view returns (uint256) {
```

```
        return data;
      }
    }
```

2.2. Viết Migration (2_deploy_simplestorage.js)

Tạo tệp **migrations/2_deploy_simplestorage.js** và dán mã sau:

JavaScript

```
// migrations/2_deploy_simplestorage.js
const SimpleStorage = artifacts.require("SimpleStorage");
module.exports = function (deployer) {
  deployer.deploy(SimpleStorage);
};
```

2.3. Cấu hình Truffle (truffle-config.js)

Mở tệp **truffle-config.js** và cấu hình mạng development sử dụng **Chain ID 5777** (một ID phổ biến cho Ganache):

JavaScript

```
// truffle-config.js (Chỉ phần networks và compilers)
module.exports = {
  networks: {
    development: {
      host: "127.0.0.1",
      port: 8545,
      network_id: 5777, // Chain ID cho Ganache
    },
  },
  compilers: {
    solc: {
      version: "0.8.0",
```

```
    }
  }
};
```

2.4. Biên dịch và Triển khai Contract

1. **Chạy Ganache:** Mở **Terminal mới** trong VS Code và khởi động Ganache với Chain ID đã cấu hình.

   Bash

   ganache --chain.chainId 5777
2. **Biên dịch Contract:** Quay lại Terminal dự án Truffle và biên dịch.

   Bash

   truffle compile
3. **Triển khai Contract:** Triển khai lên Ganache. **GHI LẠI ĐỊA CHỈ HỢP ĐỒNG (CONTRACT ADDRESS)**.

   Bash

   truffle migrate --network development

3. Cấu hình MetaMask

3.1. Cài đặt MetaMask

Cài đặt tiện ích mở rộng MetaMask trên trình duyệt của bạn (nếu chưa có) và hoàn thành các bước thiết lập ví.

3.2. Kết nối MetaMask với Ganache

1. Mở MetaMask, nhấp vào menu mạng (thường là "Ethereum Mainnet") và chọn **"Add network"** (Thêm mạng).
2. Chọn **"Add a network manually"** (Thêm mạng thủ công).
3. Nhập thông tin Ganache:
   - **Network name:** Ganache Local 5777
   - **New RPC URL:** http://127.0.0.1:8545
   - **Chain ID:** 5777

- **Currency symbol:** ETH

4. Lưu lại và **chuyển sang mạng** này. Bạn sẽ thấy **100 ETH** giả lập trong tài khoản.

4. Phát triển Front-end (HTML & Ethers.js)

4.1. Thiết lập Cấu trúc Front-end

Tạo thư mục **client/** và thêm hai tệp:

Bash

```
mkdir client
touch client/index.html client/script.js
```

4.2. Mã HTML (client/index.html)

Dán mã giao diện và tải Ethers.js:

```
HTML
<!DOCTYPE html>
<html lang="vi">
<head>
    <meta charset="UTF-8">
    <title>Simple Storage DApp</title>
    <script src="https://cdn.ethers.io/lib/ethers-5.7.0.umd.min.js"
type="application/javascript"></script>
    <script src="script.js" defer></script>
</head>
<body>
    <h1>Simple Storage DApp</h1>
    <button id="connectWallet">1. Kết nối Ví (MetaMask)</button>
    <p>Trạng thái Ví: <span id="walletStatus">Chưa kết nối</span></p>
    <hr>
    <h2>Đọc/Ghi Dữ liệu</h2>
```

```
    <button id="readData">Lấy Giá trị Hiện tại</button>
    <p>Giá trị Đã lưu: <span id="currentValue">...</span></p>
    <br>
    <input type="number" id="newValue" placeholder="Nhập số mới">
    <button id="writeData">Lưu Giá trị Mới</button>
    <p id="transactionStatus"></p>
  </body>
  </html>
```

4.3. Mã JavaScript (client/script.js)

THAY THẾ CONTRACT_ADDRESS và CONTRACT_ABI bằng thông tin của bạn.

JavaScript

```
// client/script.js
// THAY THẾ bằng địa chỉ hợp đồng của bạn từ truffle migrate
const CONTRACT_ADDRESS = "ĐỊA CHỈ HỢP ĐỒNG CỦA BẠN";
// Dán mảng ABI JSON từ build/contracts/SimpleStorage.json
const CONTRACT_ABI = [
  {"inputs": [],"name": "get","outputs": [{"internalType": "uint256","name": "","type": "uint256"}],"stateMutability": "view","type": "function"},
  {"inputs": [{"internalType": "uint256","name": "newData","type": "uint256"}],"name": "set","outputs": [],"stateMutability": "nonpayable","type": "function"}
];
let signer = null;
let simpleStorageContract = null;
// --- 1. KẾT NỐI VÍ ---
```

```
document.getElementById('connectWallet').addEventListener('click',
connectWallet);
async function connectWallet() {
   if (typeof window.ethereum === 'undefined') { return alert("Vui lòng cài đặt
MetaMask!"); }
       try {
       const provider = new ethers.providers.Web3Provider(window.ethereum);
       await provider.send("eth_requestAccounts", []);
            signer = provider.getSigner();
       simpleStorageContract = new ethers.Contract(CONTRACT_ADDRESS,
CONTRACT_ABI, signer);
       document.getElementById('walletStatus').textContent = `Đã kết nối: ${await
signer.getAddress()}`;
   } catch (error) {
       console.error("Lỗi kết nối:", error);
   }
}
// --- 2. ĐỌC DỮ LIỆU (GET) ---
document.getElementById('readData').addEventListener('click', readData);
async function readData() {
   if (!simpleStorageContract) { return alert("Kết nối ví trước!"); }
   try {
       const data = await simpleStorageContract.get();
       document.getElementById('currentValue').textContent = data.toString();
   } catch (error) {
       console.error("Lỗi khi đọc dữ liệu:", error);
   }
```

```
}
// --- 3. GHI DỮ LIỆU (SET) ---
document.getElementById('writeData').addEventListener('click', writeData);
async function writeData() {
    if (!simpleStorageContract) { return alert("Kết nối ví trước!"); }
    const newValue = document.getElementById('newValue').value;
    if (!newValue || isNaN(newValue)) { return alert("Nhập số hợp lệ."); }
    const statusElement = document.getElementById('transactionStatus');
    statusElement.textContent = "Đang gửi giao dịch... Vui lòng xác nhận trong MetaMask.";
    try {
        const tx = await simpleStorageContract.set(newValue);
        await tx.wait(); // Chờ xác nhận
        statusElement.textContent = `Giao dịch HOÀN TẤT! Giá trị mới: ${newValue}`;
        readData();
    } catch (error) {
        console.error("Lỗi giao dịch:", error);
        statusElement.textContent = `Giao dịch THẤT BẠI.`;
    }
}
```

Bước 5. Chạy DApp

1. **Chạy Ganache:** Đảm bảo nó đang hoạt động trong terminal.
2. **Chạy Front-end:** Trong VS Code, nhấp chuột phải vào **client/index.html** và chọn **"Open with Live Server"** (hoặc sử dụng máy chủ web cục bộ khác).
3. **Thử nghiệm:**
    - Nhấp vào **"Kết nối Ví"** và phê duyệt trong MetaMask.

- Nhấp vào **"Lấy Giá trị Hiện tại"** (ban đầu sẽ là 0).
- Nhập một số, nhấp vào **"Lưu Giá trị Mới"** và xác nhận giao dịch trong MetaMask.

Bài 3: Xây dựng ứng dụng (Dapp) truy xuất nguồn gốc thực phẩm với Blockchain Ethereum. Yêu cầu sử dụng Node.js, truffle,ganache, metamask, ethers.js/web3.js, scripts trong môi trường VS Code.

Cấu trúc Dự án

Trong thư mục gốc food-trace-dapp có cấu trúc như sau:

food-trace-dapp/

├── contracts/
│ └── FoodTraceability.sol
├── migrations/
├── build/
│ └── contracts/
│ └── FoodTraceability.json <-- File chứa ABI và Contract Address
├── client/
│ ├── index.html <-- Giao diện người dùng
│ ├── script.js <-- Logic kết nối Blockchain (Ethers.js)
│ └── style.css <-- CSS (Tùy chọn, để DApp đẹp hơn)
└── truffle-config.js

Thành phần	Công cụ	Vai trò
Blockchain	**Ganache**	Mạng lưới Ethereum cục bộ để phát triển và kiểm thử.
Smart Contract	**Solidity**	Hợp đồng lưu trữ logic truy xuất nguồn gốc (ví dụ: đăng ký sản phẩm, chuyển giao quyền sở hữu).

Thành phần	Công cụ	Vai trò
Development Tool	**Truffle**	Biên dịch, kiểm thử và triển khai Smart Contract.
Front-end DApp	**Node.js, HTML, JS**	Giao diện người dùng web.
Web3 Library	**Ethers.js**	Thư viện JavaScript kết nối Front-end với Smart Contract.
Wallet	**MetaMask**	Ví tiền điện tử để ký giao dịch (phải trả Gas).

I. Back-end: Smart Contract & Deployment

Bước 1: Khởi tạo Dự án & Chuẩn bị Môi trường

1. **Mở Terminal** trong VS Code và tạo dự án:

   mkdir food-trace-dapp

   cd food-trace-dapp

   truffle init

2. **Khởi động Ganache:** Trong một terminal khác, khởi động Ganache với Chain ID cố định (ví dụ: 5777).

   ganache --chain.chainId 5777

Bước 2: Thiết kế và Viết Smart Contract (Solidity)

Tạo tệp FoodTraceability.sol trong thư mục contracts/.

Đây là hợp đồng cơ bản theo dõi sản phẩm (Product) qua các giai đoạn (Stage).

Solidity

// contracts/FoodTraceability.sol

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract FoodTraceability {

```
    // Cấu trúc dữ liệu cho một sản phẩm
    struct Product {
        uint256 id;
        string name;
        address owner;
        string location; // Vị trí hiện tại
        string history;  // Lịch sử giao dịch/chuyển giao
    }
    // Map ID sản phẩm với thông tin sản phẩm
    mapping(uint256 => Product) public products;
    uint256 public nextProductId = 1;
   // Sự kiện được phát ra khi sản phẩm được đăng ký
    event ProductRegistered(uint256 id, address indexed owner, string name);
    // Sự kiện được phát ra khi sản phẩm được chuyển giao/cập nhật
    event ProductUpdated(uint256 id, address indexed newOwner, string
newLocation, string timestamp);
    // 1. Hàm Đăng ký Sản phẩm (Chỉ người gọi mới có quyền sở hữu ban đầu)
    function registerProduct(string memory _name, string memory _initialLocation)
public {
        uint256 productId = nextProductId;
        products[productId] = Product(
            productId,
            _name,
            msg.sender, // Người gọi hàm là chủ sở hữu ban đầu
            _initialLocation,
            string(abi.encodePacked("Registered at ", _initialLocation))
        );
```

```
        nextProductId++;
        emit ProductRegistered(productId, msg.sender, _name);
    }
    // 2. Hàm Cập nhật (Chuyển giao) Sản phẩm (Chỉ Chủ sở hữu hiện tại mới được gọi)
    function updateProduct(uint256 _id, address _newOwner, string memory _newLocation, string memory _timestamp) public {
        require(products[_id].owner == msg.sender, "Chi co chu so huu moi co quyen cap nhat.");
        Product storage p = products[_id];
            // Cập nhật lịch sử
        p.history = string(abi.encodePacked(p.history, " -> Transfer to ", _newLocation, " on ", _timestamp));
        // Cập nhật chủ sở hữu và vị trí
        p.owner = _newOwner;
        p.location = _newLocation;
        emit ProductUpdated(_id, _newOwner, _newLocation, _timestamp);
    }
    // 3. Hàm Truy xuất thông tin sản phẩm (Call)
    function getProduct(uint256 _id) public view returns (uint256, string memory, address, string memory, string memory) {
        Product memory p = products[_id];
        return (p.id, p.name, p.owner, p.location, p.history);
    }
}
```

Bước 3: Biên dịch và Triển khai (Deploy)

1. **Viết Migration:** Tạo tệp 2_deploy_foodtraceability.js trong thư mục migrations/.

   JavaScript

   // migrations/2_deploy_foodtraceability.js

   const FoodTraceability = artifacts.require("FoodTraceability");

   module.exports = function (deployer) {

   deployer.deploy(FoodTraceability);

   };
2. **Biên dịch:**

   truffle compile
3. **Triển khai:**

   truffle migrate --network development

   Ghi lại **Contract Address** từ kết quả triển khai và **ABI** từ tệp build/contracts/FoodTraceability.json.

II. Front-end: Thiết kế Giao diện DApp

Bước 4: Thiết lập Giao diện HTML và CSS

1. **Tạo thư mục client:**

   mkdir client
2. **Tệp client/index.html:**
   - Sử dụng CDN của **Ethers.js** (thư viện web3 hiện đại).
   - Kết nối với script.js và style.css.

HTML

<!DOCTYPE html>

<html lang="vi">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

```
    <title>DApp Truy Xuất Nguồn Gốc Thực Phẩm</title>
    <link rel="stylesheet" href="style.css">
    <script src="https://cdn.ethers.io/lib/ethers-5.7.0.umd.min.js" type="application/javascript"></script>
    <script src="script.js" defer></script>
</head>
<body>
    <div class="container">
        <h1>
[Image of a magnifying glass over a piece of food]
 DApp Truy Xuất Nguồn Gốc Thực Phẩm</h1>
        <section class="card" id="wallet-section">
            <h2>1. Kết nối MetaMask</h2>
            <button id="connectWallet" class="btn primary">Kết nối Ví</button>
            <p id="walletStatus" class="status-text"></p>
        </section>
        <section class="card">
            <h2>2. Đăng ký Sản phẩm (Giao dịch)</h2>
            <input type="text" id="regName" placeholder="Tên sản phẩm (ví dụ: Cà chua)" required>
            <input type="text" id="regLocation" placeholder="Vị trí ban đầu (ví dụ: Nông trại A)" required>
            <button id="registerBtn" class="btn secondary">Đăng ký Sản phẩm</button>
            <p id="regStatus" class="status-text"></p>
        </section>
        <section class="card">
```

```
        <h2>3. Chuyển giao / Cập nhật Sản phẩm (Giao dịch)</h2>
        <input type="number" id="transferId" placeholder="ID Sản phẩm" required>
        <input type="text" id="newOwnerAddr" placeholder="Địa chỉ Chủ sở hữu Mới (0x...)" required>
        <input type="text" id="transferLocation" placeholder="Vị trí chuyển đến (ví dụ: Kho lạnh B)" required>
        <button id="transferBtn" class="btn secondary">Chuyển giao</button>
        <p id="transferStatus" class="status-text"></p>
      </section>
      <section class="card">
        <h2>4. Truy xuất Thông tin (Đọc)</h2>
        <input type="number" id="traceId" placeholder="Nhập ID Sản phẩm để truy xuất" required>
        <button id="traceBtn" class="btn primary">Truy xuất</button>
        <pre id="traceResult" class="trace-output">Kết quả sẽ hiển thị ở đây...</pre>
      </section>
    </div>
</body>
</html>
```

3. **Tệp client/style.css (CSS cơ bản):**

CSS

```
/* client/style.css */
body {
    font-family: Arial, sans-serif;
    background-color: #f4f7f9;
```

```
    color: #333;
    line-height: 1.6;
    margin: 0;
    padding: 20px;
}
.container {
    max-width: 800px;
    margin: 0 auto;
    background: #fff;
    padding: 30px;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0,0,0,0.1);
}
h1 {
    color: #007bff;
    text-align: center;
    margin-bottom: 30px;
}
.card {
    background: #f9f9f9;
    padding: 20px;
    border-radius: 6px;
    margin-bottom: 20px;
    border-left: 5px solid #007bff;
}
h2 {
    color: #333;
```

```
    border-bottom: 1px solid #ddd;
    padding-bottom: 10px;
    margin-top: 0;
}
input[type="text"], input[type="number"] {
    width: 100%;
    padding: 10px;
    margin: 8px 0;
    display: inline-block;
    border: 1px solid #ccc;
    border-radius: 4px;
    box-sizing: border-box;
}
.btn {
    background-color: #007bff;
    color: white;
    padding: 10px 15px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 16px;
    margin-top: 10px;
}
.btn:hover {
    opacity: 0.9;
}
.secondary {
```

```
    background-color: #28a745; /* Màu xanh lá cho hành động Transaction */
}
.secondary:hover {
    background-color: #218838;
}
.status-text {
    margin-top: 10px;
    font-weight: bold;
    color: #dc3545;
}
.trace-output {
    background: #e9ecef;
    padding: 15px;
    border-radius: 4px;
    border: 1px solid #ced4da;
    white-space: pre-wrap;
    word-wrap: break-word;
    font-size: 0.9em;
}
```

III. Front-end: Logic Kết nối Blockchain (Ethers.js)

Bước 5: Viết Mã JavaScript (client/script.js)

Đây là phần quan trọng nhất, nơi Ethers.js được sử dụng để gọi các hàm **view/call** (đọc dữ liệu không tốn gas) và **transaction** (ghi dữ liệu, tốn gas và cần ký bằng MetaMask).

BẠN PHẢI THAY THẾ HAI HẰNG SỐ DƯỚI ĐÂY BẰNG DỮ LIỆU CỦA MÌNH

JavaScript

```
// client/script.js
// LẤY TỪ build/contracts/FoodTraceability.json SAU KHI DEPLOY
const CONTRACT_ADDRESS = "0xe78A0F7E598Cc8b0Bb87894B0F60dD2a88d6a8Ab"; // Thay bằng địa chỉ của bạn
const CONTRACT_ABI = [
  // Dán mảng ABI (toàn bộ nội dung của khóa "abi" trong file .json) vào đây
  // Vì mảng ABI rất dài, bạn cần sao chép nó một cách cẩn thận.
  // Ví dụ (chỉ là ví dụ):
  {
    "inputs": [
      { "internalType": "string", "name": "_name", "type": "string" },
      { "internalType": "string", "name": "_initialLocation", "type": "string" }
    ],
    "name": "registerProduct",
    "outputs": [],
    "stateMutability": "nonpayable",
    "type": "function"
  },
  {
    "inputs": [ { "internalType": "uint256", "name": "_id", "type": "uint256" } ],
    "name": "getProduct",
    "outputs": [
      { "internalType": "uint256", "name": "", "type": "uint256" },
      { "internalType": "string", "name": "", "type": "string" },
      { "internalType": "address", "name": "", "type": "address" },
      { "internalType": "string", "name": "", "type": "string" },
```

```
      { "internalType": "string", "name": "", "type": "string" }
    ],
    "stateMutability": "view",
    "type": "function"
  },
  // ... (và các function, event khác)
];
let provider = null;
let signer = null;
let traceContract = null;
// ============================================================
// A. KẾT NỐI VÍ VÀ KHỞI TẠO CONTRACT
// ============================================================
document.getElementById('connectWallet').addEventListener('click',
connectWallet);

async function connectWallet() {
    if (typeof window.ethereum === 'undefined') {
        document.getElementById('walletStatus').textContent = "Lỗi: Vui lòng cài đặt
MetaMask!";
        return;
    }
    try {
        // Yêu cầu MetaMask kết nối
        provider = new ethers.providers.Web3Provider(window.ethereum);
        await provider.send("eth_requestAccounts", []);
        signer = provider.getSigner();
```

```
        // Khởi tạo Contract với Signer (cho phép ký giao dịch)
        traceContract = new ethers.Contract(CONTRACT_ADDRESS, CONTRACT_ABI, signer);
        const currentAddress = await signer.getAddress();
        document.getElementById('walletStatus').innerHTML = 'Đã kết nối: ${currentAddress}';
    } catch (error) {
        console.error("Lỗi kết nối:", error);
        document.getElementById('walletStatus').textContent = "Lỗi kết nối ví. Hãy kiểm tra mạng Ganache và MetaMask.";
    }
}

// ============================================================
// B. ĐĂNG KÝ SẢN PHẨM (TRANSACTION - GHI DỮ LIỆU)
// ============================================================
document.getElementById('registerBtn').addEventListener('click', registerProduct);
async function registerProduct() {
    if (!traceContract || !signer) return alert("Vui lòng kết nối ví trước!");
        const name = document.getElementById('regName').value;
    const location = document.getElementById('regLocation').value;
    const statusEl = document.getElementById('regStatus');
    if (!name || !location) return alert("Vui lòng nhập đầy đủ thông tin.");
    try {
        statusEl.textContent = "Đang gửi giao dịch đăng ký... (Vui lòng xác nhận trên MetaMask)";
```

```
        // Gọi hàm registerProduct trên Smart Contract
        const tx = await traceContract.registerProduct(name, location);
        // Chờ giao dịch được khai thác và xác nhận
        await tx.wait();
        // Trích xuất ID sản phẩm từ Event (Phần phức tạp hơn, có thể bỏ qua bước này nếu khó)
        // Cách đơn giản là chỉ báo thành công:
        statusEl.innerHTML = `Đăng ký HOÀN TẤT. Vui lòng truy xuất ID 1, 2, 3... để kiểm tra.`;
    } catch (error) {
        console.error("Lỗi đăng ký:", error);
        statusEl.textContent = `Đăng ký THẤT BẠI: ${error.message}. (Có thể do bạn từ chối giao dịch)`;
    }
}

// ============================================================
// C. CHUYỂN GIAO SẢN PHẨM (TRANSACTION - GHI DỮ LIỆU)
// ============================================================
document.getElementById('transferBtn').addEventListener('click', transferProduct);
async function transferProduct() {
    if (!traceContract || !signer) return alert("Vui lòng kết nối ví trước!");
    const id = document.getElementById('transferId').value;
    const newOwner = document.getElementById('newOwnerAddr').value;
    const location = document.getElementById('transferLocation').value;
    const statusEl = document.getElementById('transferStatus');
```

```
    const timestamp = new Date().toLocaleString('vi-VN');
    if (!id || !newOwner || !location) return alert("Nhập đủ ID, địa chỉ chủ sở hữu mới, và vị trí.");
    try {
        statusEl.textContent = "Đang gửi giao dịch chuyển giao... (Vui lòng xác nhận trên MetaMask)";
        // Gọi hàm updateProduct
        const tx = await traceContract.updateProduct(id, newOwner, location, timestamp);
        await tx.wait();
        statusEl.innerHTML = `Chuyển giao HOÀN TẤT! Sản phẩm **#${id}** đã chuyển đến **${location}**.`;
    } catch (error) {
        console.error("Lỗi chuyển giao:", error);
        // Lỗi này thường do người gọi không phải chủ sở hữu (xem require trong Solidity)
        statusEl.textContent = `❌ Chuyển giao THẤT BẠI: ${error.message}. (Kiểm tra xem bạn có phải là chủ sở hữu hiện tại không?)`;
    }
}

// ==========================================================
// D. TRUY XUẤT THÔNG TIN (CALL - ĐỌC DỮ LIỆU)
// ==========================================================
document.getElementById('traceBtn').addEventListener('click', traceProduct);
async function traceProduct() {
    if (!traceContract) return alert("Vui lòng kết nối ví trước!");
```

```
    const id = document.getElementById('traceId').value;
  const resultEl = document.getElementById('traceResult');
        if (!id) return alert("Vui lòng nhập ID sản phẩm.");
  try {
    // Gọi hàm getProduct (Call) - Tương tự như gọi API, không cần ký, không tốn gas
    // Sử dụng traceContract.connect(provider) để chỉ đọc (view)
    const readOnlyContract = traceContract.connect(provider);
        const [productId, name, owner, location, history] = await readOnlyContract.getProduct(id);
    resultEl.textContent = `
      ==========================================
      ID Sản phẩm: ${productId.toString()}
      Tên Sản phẩm: ${name}
      ----------------------------------------
      Chủ sở hữu hiện tại: ${owner}
      Vị trí hiện tại: ${location}
      ==========================================
      Lịch sử Nguồn gốc (Trace History):
      ${history}
      ==========================================
    `;   } catch (error) {
    console.error("Lỗi truy xuất:", error);
    resultEl.textContent = ` Truy xuất THẤT BẠI: ID không hợp lệ hoặc lỗi kết nối.`;
  }
}
```

V. Hoàn thiện và Kiểm tra

Bước 6: Chạy DApp

1. **Cài đặt Live Server:** Trong VS Code, cài tiện ích mở rộng **Live Server**.
2. **Chạy Server:** Mở tệp client/index.html và nhấn chuột phải, chọn **"Open with Live Server"**. Trình duyệt sẽ mở DApp của bạn.

Bước 7: Quy trình Thử nghiệm

1. **Kết nối Ví:** Nhấp vào **"Kết nối Ví"**. MetaMask sẽ hiện lên yêu cầu kết nối.
2. **Đăng ký (Account 1):**
    - Tên: Bắp Cải
    - Vị trí: Nông trại Đà Lạt
    - Nhấp **"Đăng ký Sản phẩm"** và ký giao dịch trên MetaMask.
    - Giả sử sản phẩm được gán **ID 1**.
3. **Truy xuất (Call):**
    - Nhập **ID 1**.
    - Nhấp **"Truy xuất"**. Kết quả sẽ hiển thị lịch sử ban đầu.
4. **Chuyển giao (Account 1):**
    - ID: **1**
    - Địa chỉ Chủ sở hữu Mới: Dán địa chỉ của **Account 2** từ MetaMask (chuyển sang Account 2 và sao chép địa chỉ).
    - Vị trí chuyển đến: Chợ Đầu Mối
    - Nhấp **"Chuyển giao"** và ký giao dịch.
5. **Kiểm tra lại (Call):**
    - Truy xuất lại **ID 1**. Bạn sẽ thấy Chủ sở hữu hiện tại là Account 2, và lịch sử đã được nối thêm.

Bài 4: Hãy triển khai bài 3 lên mạng Internet

Để triển khai ứng dụng truy xuất nguồn gốc thực phẩm lên một **mạng Ethereum thực sự** (thường là một Testnet như **Sepolia** hoặc **Holesky** trước khi lên Mainnet), cần thực hiện các thay đổi chính sau trong cấu hình Truffle và quá trình triển khai:

1. Chuẩn bị Tài khoản và ETH Thử nghiệm

1.1. Lấy ETH Thử nghiệm (Testnet Ether)

Không giống Ganache, mạng Testnet yêu cầu bạn sử dụng ETH thử nghiệm để trả phí gas cho việc triển khai và các giao dịch.

- Đảm bảo ví **MetaMask** của bạn đang chuyển sang mạng **Sepolia** hoặc **Holesky**.
- Truy cập một trang **Faucet** (ví dụ: faucet.sepolia.dev hoặc các faucet uy tín khác) và dán địa chỉ ví MetaMask của bạn để nhận một lượng nhỏ ETH thử nghiệm miễn phí.

1.2. Cài đặt HD Wallet Provider

Bạn cần một thư viện để quản lý việc ký giao dịch bằng **Cụm từ Khôi phục Bí mật (Mnemonic)** hoặc **Khóa riêng tư (Private Key)** của ví MetaMask.

Bash

```
npm install @truffle/hdwallet-provider
```

2. Cấu hình Truffle cho Testnet

Bạn cần chỉnh sửa tệp **truffle-config.js** để thêm cấu hình mạng Sepolia và sử dụng HD Wallet Provider.

- **Lấy Mnemonic:** Lấy Mnemonic (12 từ bí mật) từ MetaMask. **LƯU Ý:** Tuyệt đối không sử dụng Mnemonic của ví Mainnet có tài sản thật. Đây phải là Mnemonic của ví thử nghiệm mới hoặc ví bạn sử dụng cho phát triển.
- **Lấy Infura/Alchemy Project ID:** Đăng ký tài khoản trên **Infura** hoặc **Alchemy** để có một **RPC endpoint** kết nối với mạng Sepolia.

JavaScript

```
// truffle-config.js
```

```
const HDWalletProvider = require('@truffle/hdwallet-provider');
//  THAY THẾ bằng Mnemonic của ví Sepolia/Testnet của bạn
const MNEMONIC = "từ-bí-mật-của-bạn-từ-1-đến-12";
// THAY THẾ bằng Project ID từ Infura hoặc Alchemy
const INFURA_PROJECT_ID = "ID-Project-của-bạn";
module.exports = {
  // ... (giữ nguyên compilers)
  networks: {
    // Cấu hình Ganache cục bộ vẫn giữ nguyên cho việc phát triển
    development: {
      host: "127.0.0.1",
      port: 8545,
      network_id: 5777,
    },
    // Cấu hình Mạng Thử nghiệm Sepolia
    sepolia: {
      provider: () => new HDWalletProvider(
        MNEMONIC,
        `https://sepolia.infura.io/v3/${INFURA_PROJECT_ID}`
      ),
      network_id: 11155111, // Chain ID của Sepolia
      gas: 5500000,         // Giới hạn Gas
      confirmations: 2,     // Số block chờ xác nhận
      timeoutBlocks: 200,
      skipDryRun: true      // Bỏ qua chạy thử
    },
    // ... (có thể thêm Holesky, Mainnet)
```

```
  },
  // ... (giữ nguyên compilers)
};
```

3. Triển khai lên Testnet

Sử dụng lệnh truffle migrate nhưng trỏ đến mạng **sepolia**.

truffle migrate --network sepolia

- **Kết quả:** Truffle sẽ sử dụng tài khoản ví của bạn, kết nối qua Infura/Alchemy đến mạng Sepolia, ký giao dịch và triển khai hợp đồng của bạn. Quá trình này sẽ tốn ETH thử nghiệm.

4. Cập nhật Front-end (DApp)

Sau khi triển khai thành công lên Sepolia:

1. **Lấy Contract Address mới:** Ghi lại địa chỉ hợp đồng mới đã được triển khai trên Sepolia.
2. **Cập nhật script.js:** Thay thế **CONTRACT_ADDRESS** cũ (của Ganache) bằng địa chỉ mới.
3. **Cập nhật MetaMask:** Hướng dẫn người dùng chuyển MetaMask sang mạng **Sepolia**.

Lúc này, Front-end sẽ kết nối với MetaMask và MetaMask sẽ gửi các giao dịch đến mạng Sepolia thông qua các RPC Endpoint công cộng.

BÀI TẬP

Nghiên cứu làm chủ 15 dự án Blockchain theo link sau:

Top 15 Blockchain Projects With Source Code [2023 Update] - InterviewBit

Yêu cầu: Nộp báo cáo 15 dự án bài tập.