**Uploading Files to a Distributed Storage in a Blockchain**

The ERC-721 is a standard for creating an NFT. Any unique thing can be an NFT. The power of NFT is in verifiability.

Tools Needed

1. Editor – Notepad++/Remix
2. IPFS
3. Ganache
4. Truffle
5. NodeJS
6. Pinata

**NodeJS**

Install NodeJS from <https://nodejs.org/en/download>

Check installation

>node -v

**IPFS**

IPFS Desktop can be installed from <https://docs.ipfs.tech/install/ipfs-desktop>. Select .exe file for Windows.

**Ganache**

Install Ganache for Windows from <https://trufflesuite.com/ganache/>. Follow the steps in setup. Keep it up and running.

**Truffle**

>npm install -g truffle

Check installation

>truffle version

**Pinata**

Sign up to the Pinata Cloud <https://www.pinata.cloud/>

Click on API Keys🡺New Key 🡺Select API Endpoint Access🡺pinFileToIPFS🡺Create Key

Copy the Generated API Keys to a File.

**Writing the Smart Contract**

Create a Folder called “myAsset”

>mkdir myAsset

Change to myAsset directory

>cd myAsset

Initialize project directory

>npm init -y

Use Truffle to initialize the smart contract

>truffle init

Install OpenZeppelin Library

>npm install @openzeppelin/contracts

Creating the NFT Token File

>type nul >> contracts/myToken.sol

Writing code to the myToken.sol File:

pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";

import "@openzeppelin/contracts/utils/Counters.sol";

contract UniqueAsset is ERC721URIStorage {

using Counters for Counters.Counter;

Counters.Counter private \_tokenIds;

mapping(string => uint8) hashes;

constructor() public ERC721("UniqueAsset", "UNA") {}

function awardItem(address recipient, string memory hash, string memory metadata) public returns (uint256)

{

require(hashes[hash] != 1);

hashes[hash] = 1;

\_tokenIds.increment();

uint256 newItemId = \_tokenIds.current();

\_mint(recipient, newItemId);

\_setTokenURI(newItemId, metadata);

return newItemId;

}

}

Compile Contract and Deploy it to Ganache:

>truffle compile

Deploy

>truffle migrate

**Uploading File to IPFS**

We will use Pinata to upload our file to IPFS.

**Installing dependencies:**

npm i axios form-data

**Create a file uploadFile.js**

const pinataApiKey = "APIKEY";

const pinataSecretApiKey = "SECRETAPIKEY";

const axios = require("axios");

const fs = require("fs");

const FormData = require("form-data");

const pinFileToIPFS = async () => {

const url = `https://api.pinata.cloud/pinning/pinFileToIPFS`;

let data = new FormData();

data.append("file", fs.createReadStream("D:\\mySpecialAsset\\contracts\\picJSON.json"));

const res = await axios.post(url, data, {

maxContentLength: "Infinity",

headers: {

"Content-Type": `multipart/form-data; boundary=${data.\_boundary}`,

pinata\_api\_key: pinataApiKey,

pinata\_secret\_api\_key: pinataSecretApiKey,

},

});

console.log(res.data);

};

pinFileToIPFS();

**Run Script in Terminal**

>node uploadFile.js

Upon successful upload we get the hash

D:\mySpecialAsset\contracts>node uploadFile.js

{

IpfsHash: 'QmPg2oP8XMBmbiEq764hzryZVCk673a5eRxGBKVpJ8bar1',

PinSize: 119,

Timestamp: '2023-04-01T16:17:09.843Z',

isDuplicate: true

}

**Create JSON File for your data:**

{

"name":"PPchaDP",

"hash": "QmQGxRdVcjxKcdBZZn3nj2zctQYtx44gke7rUXUgrQLjqG",

"by": "Arti Bansode"

}

**Access your IPFS file using**

https://gateway.pinata.cloud/ipfs/HASHOFYOURFILE

<https://gateway.pinata.cloud/ipfs/QmPg2oP8XMBmbiEq764hzryZVCk673a5eRxGBKVpJ8bar1>