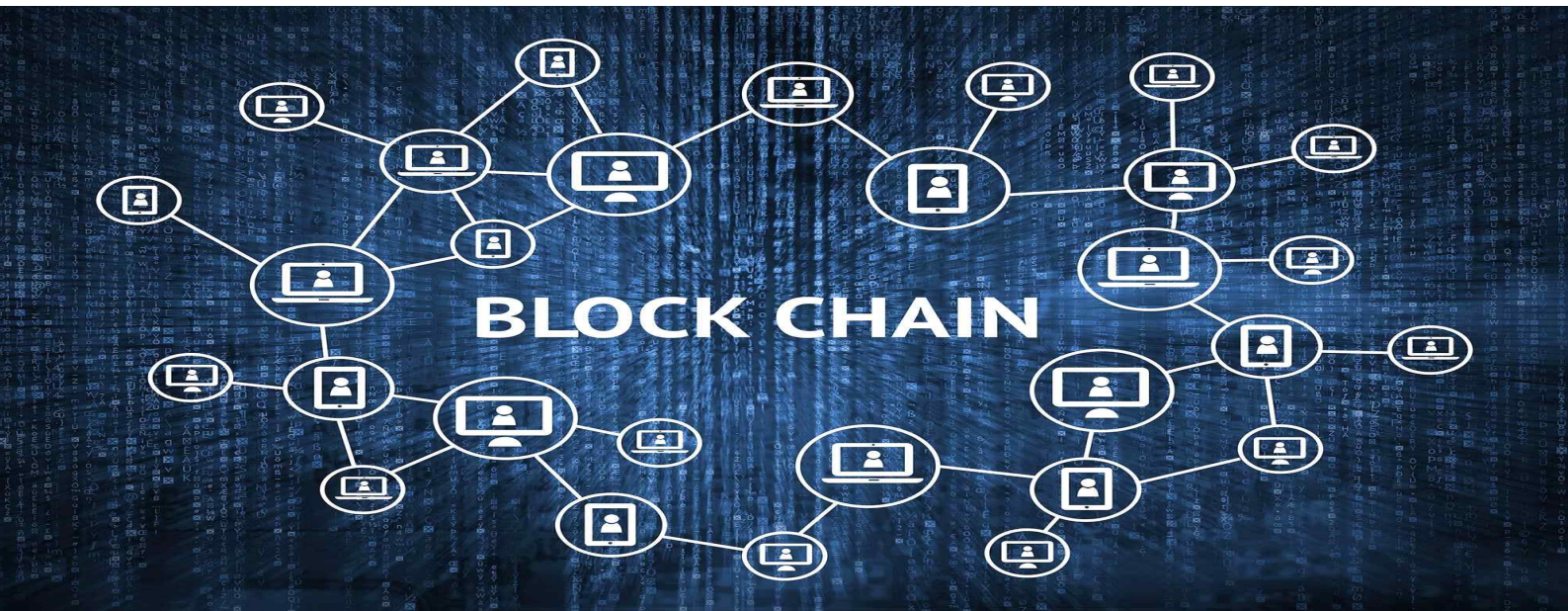


# BlockChain with NFT



## R E P O R T

과목명	캡스톤디자인-sw
담당교수	박용범 교수님
팀	팀장 소프트웨어학과 32181141 김창오 팀원 소프트웨어학과 32144794 최원제 팀원 소프트웨어학과 32180217 구선우 팀원 소프트웨어학과 32180086 강은솔

# 차 례

## 01 NFT 직접 생성해보기 03

IPFS 이해	03
Pinata 활용	04
Mytoken 생성	06

## 02 Hardhat 활용 08

Hardhat이란?	08
Install to Compile	09

## 03 실습 결론 및 소감 16

# 1. NFT 직접 생성해보기

## (1) IPFS 이해

IPFS는 InterPlanetary File System의 약자로, 분산형 파일 시스템에 데이터를 저장하고 공유하기 위한 프로토콜이며, 중앙화된 서버가 없는 파일 공유 P2P 네트워크이다.

IPFS는 파일을 블록 단위로 분할한 후 각 블록을 해싱한다. Merkle Directed Acyclic Graph (DAG) 구조로 파일을 구성한다. 이 구조는 파일의 각 블록을 그 이전의 블록의 해시 값으로 참조하는 구조이다. 이러한 블록 조합이 끝나면 이를 해싱하여 해시 값을 파일의 루트 해시(CID)로 사용한다. 그 다음 파일의 각 블록을 네트워크에 분산하여 저장한다. 저장한 파일을 찾을 때는 CID를 검색해 블록을 찾는다.

이처럼 IPFS는 해싱을 통해 고유한 식별자를 생성하여 파일의 고유성이 보장되며, 각 파일에 대한 고유한 주소도 만들 수 있다. 이러한 특성이 NFT의 고유성과 잘 맞아 NFT를 생성할 때 파일을 저장하는 형태로 적합하다. 또한 중앙화된 서버에 의존하지 않고 분산된 저장소를 이용하기 때문에 파일의 안정성이 향상되고, 비용도 줄어든다는 장점이 있다.

### IPFS와 HTTP 파일시스템의 차이

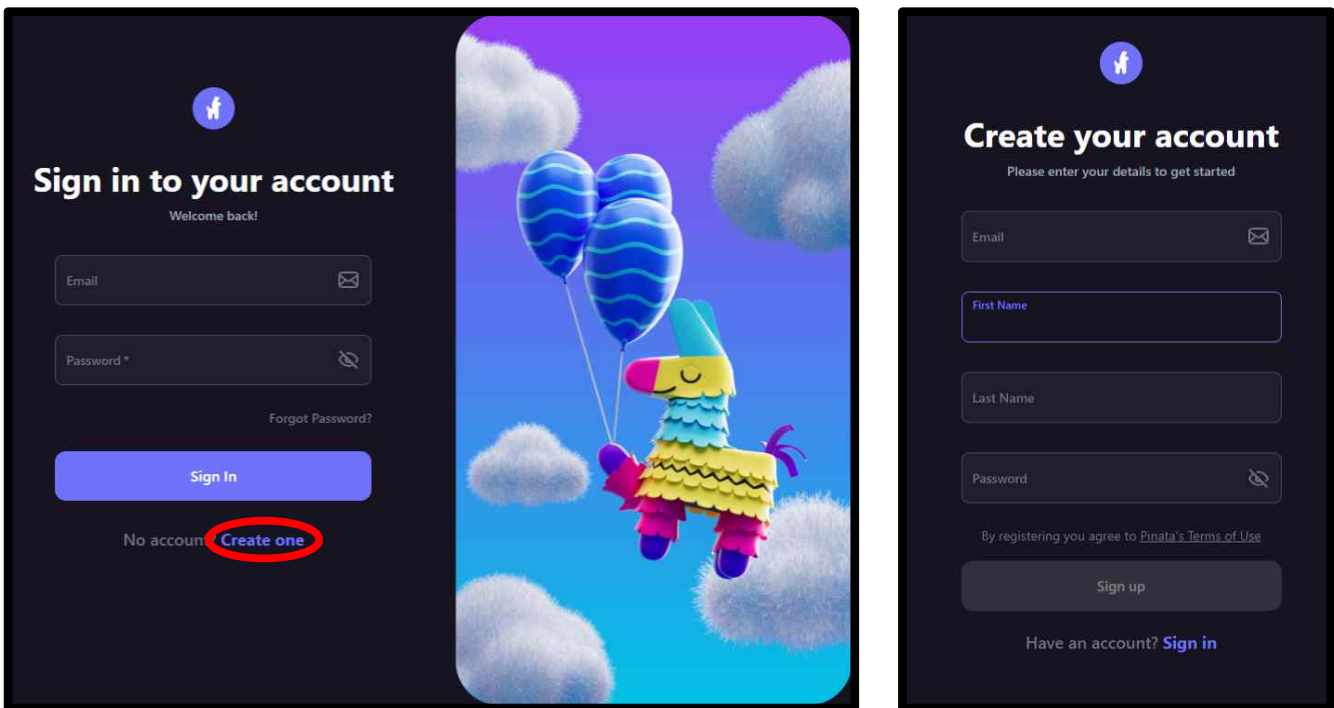
IPFS	HTTP
<ul style="list-style-type: none"> <li>· 파일의 고유한 값을 호출</li> <li>· 분산된 여러 조각의 파일을 하나로 조합</li> </ul>	<ul style="list-style-type: none"> <li>· 파일의 주소를 호출</li> <li>· 전체 파일을 한번에 가져옴</li> </ul>

## (2) Pinata 활용

IPFS를 이용해 파일을 저장하는 방법은 여러가지가 있는데 이번에 우리가 사용할 것은 Pinata이다. Pinata는 IPFS를 사용해 파일을 저장하고 관리해주는 서비스를 제공한다. 별도의 어려운 과정없이 가지고 있는 파일을 업로드하면 그 파일을 IPFS 형태로 저장해주기 때문에 사용하기 매우 간단하다.

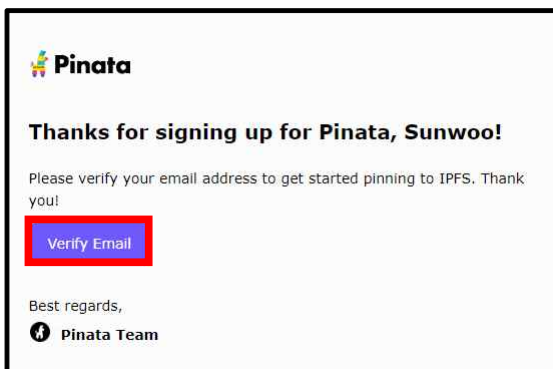
아래의 과정을 통해 Pinata를 활용해볼 수 있다.

1. Pinata 웹페이지를 검색해 들어가 Get started나 Log in 버튼을 눌러 로그인 화면에 들어간 후 Create one을 클릭해 회원가입을 진행한다.

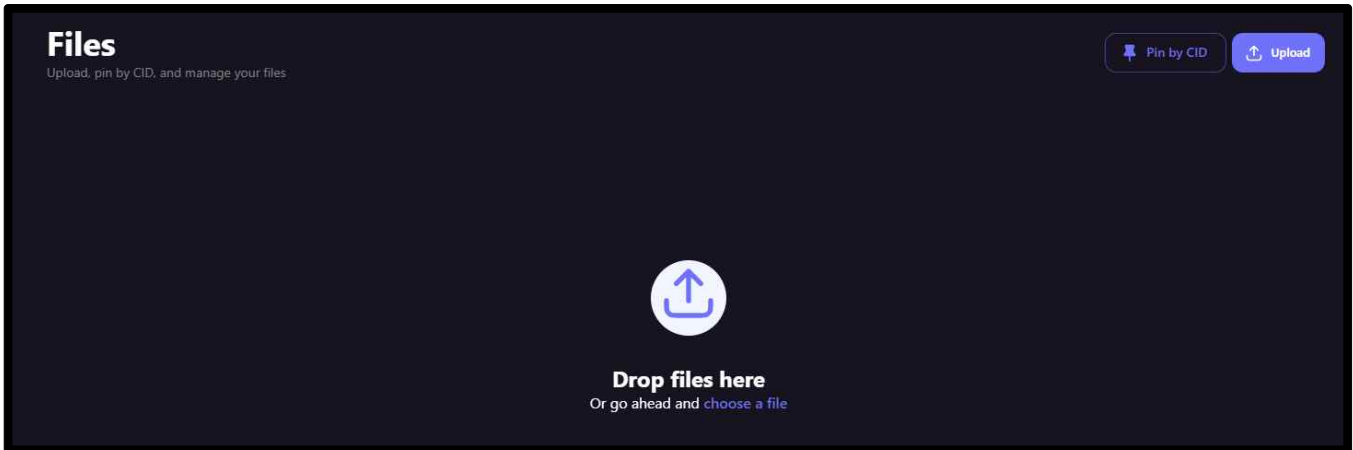


2. 입력한 이메일에 “Let's Confirm Your Pinata Account”이란 메일이 날아온다.

Verify Email 버튼을 누르면 가입이 완료된다.



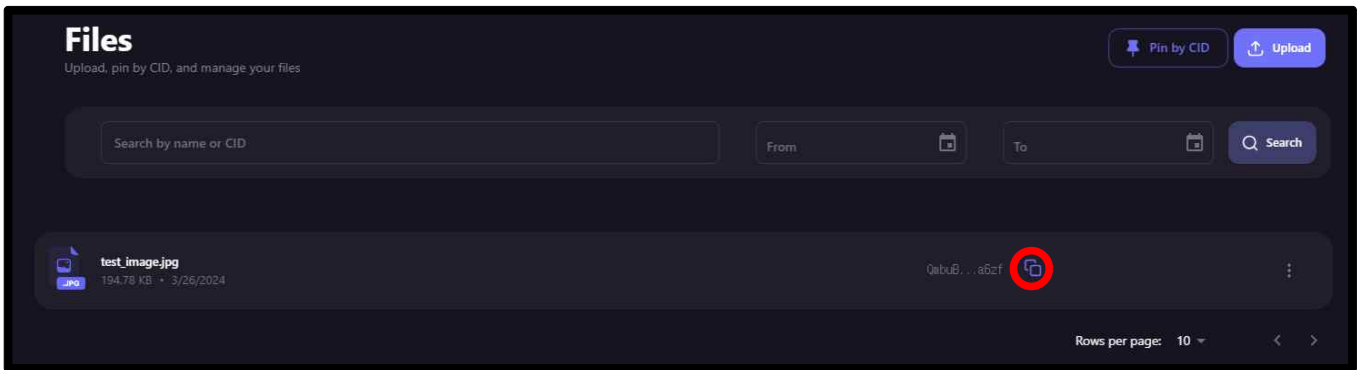
3. 로그인하면 아래와 같은 화면이 나오는데 업로드 버튼을 누르거나 가운데 아이콘으로 파일을 드래그하여 파일을 IPFS로 저장할 수 있다.



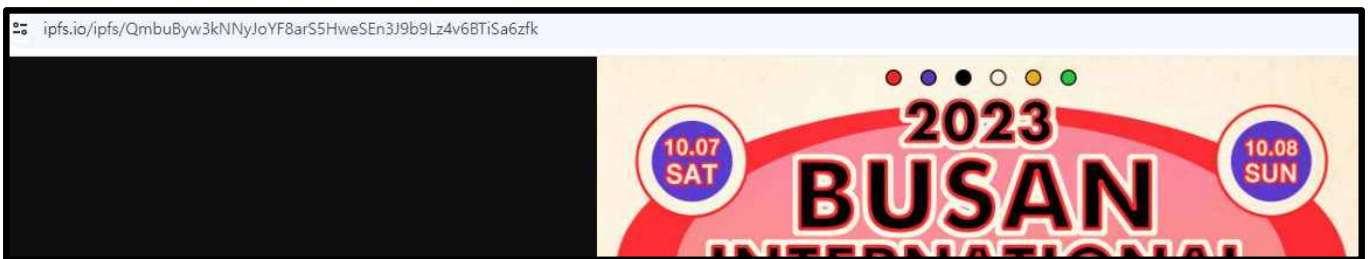
### (3) MyToken 생성

pinata를 이용하여 직접 NFT를 생성해보자.

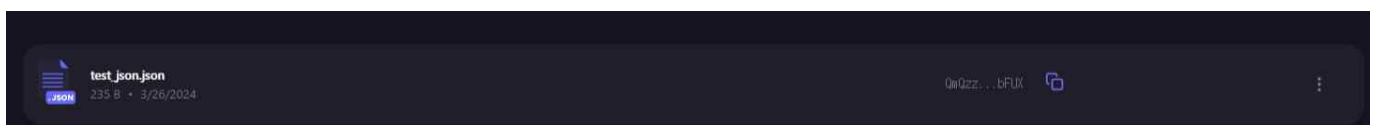
1. pinata에 NFT로 만들 이미지 파일을 업로드한다.



2. 위에 표시된 버튼을 눌러 해시 값을 복사한다. “ipfs.io/ipfs/” 뒤에 복사된 해시값을 붙여넣어 그곳으로 가면 저장된 파일을 볼 수 있다.



3. 해당 이미지의 metadata를 가진 json 파일을 만든 후 피나타에 업로드한다.



같은 방식으로 업로드된 json 파일 확인



4. 1주차에 배웠던 방법 그대로 해당 이미지 Mint

## 5. Opensea로 생성된 NFT 확인

The screenshot shows an NFT listing on the OpenSea platform. The main image is a vibrant poster for the '2023 BUSAN INTERNATIONAL ROCK FESTIVAL'. The poster features a central illustration of a rock band performing on a stage, surrounded by various musical instruments and festival-related graphics. Text on the poster includes '10.07 SAT', '10.08 SUN', '2023 BUSAN INTERNATIONAL ROCK FESTIVAL', and 'SAMNAK PARK'. Below the poster, there are sections for 'Description', 'About MyToken', and 'Details'. To the right of the poster, there are tabs for 'Price History', 'Listings', and 'Offers'. The 'Price History' tab is active, showing a message: 'No events have occurred yet. Check back later.' The 'Listings' and 'Offers' tabs also show messages: 'No listings yet' and 'No offers yet' respectively. At the bottom, there is a table with columns for 'Event', 'Price', 'From', 'To', and 'Date'. The table contains one row with the following data: 'Mint', '0.000000', 'you', and '10m ago'.

2023 Busan international rock festival #1

Owned by you

Price History

No events have occurred yet.  
Check back later.

Listings

Offers

No offers yet.

Description

By you  
This is a poster of Busan international rock festival

About MyToken

Details

Item Activity

Event	Price	From	To	Date
Mint	0.000000	you		10m ago



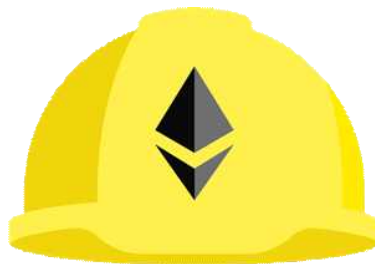
## 2. Hardhat 활용

### (1) Hardhat이란?

Hardhat은 Solidity로 작성된 이더리움 스마트 컨트랙트를 구축하고 테스트하기 위한 개발 프레임워크이며 블록체인 개발 분야에서 상당한 인기를 얻고 있다.

Hardhat은 주요한 기능은 다음과 같다.

1. Solidity 지원: 이더리움 블록체인을 위한 가장 널리 사용되는 스마트 컨트랙트 프로그래밍 언어인 Solidity와 원활하게 통합된다.
2. 테스트 기능: 스마트 컨트랙트에 대한 테스트를 작성하고 실행하기 위한 강력한 테스트 프레임워크를 제공한다.
3. 고급 디버깅: 내장된 이더리움 네트워크, 디버깅을 위한 콘솔, 테스트 환경과 같은 강력한 도구를 제공합니다.
4. TypeScript 통합: 개발자는 TypeScript로 이더리움 스마트 컨트랙트를 작성할 수 있어, 타입의 안정성과 향상된 개발 경험을 제공할 수 있다.



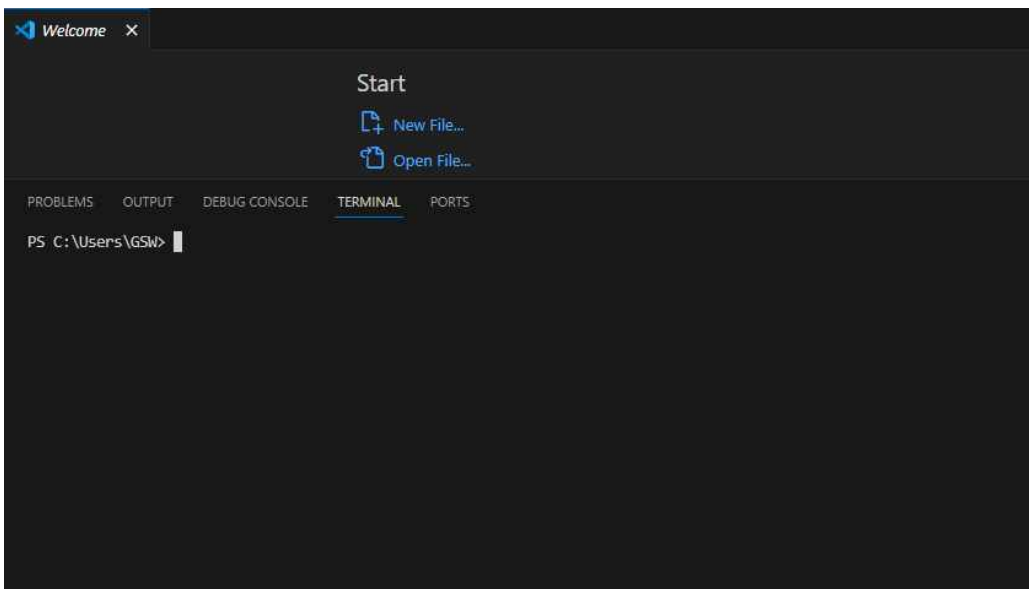
이번 시간에는 Hardhat을 설치하고 파일을 컴파일하는 과정까지를 배웠다. 아래에서 그 과정을 자세히 다룰 것이다.



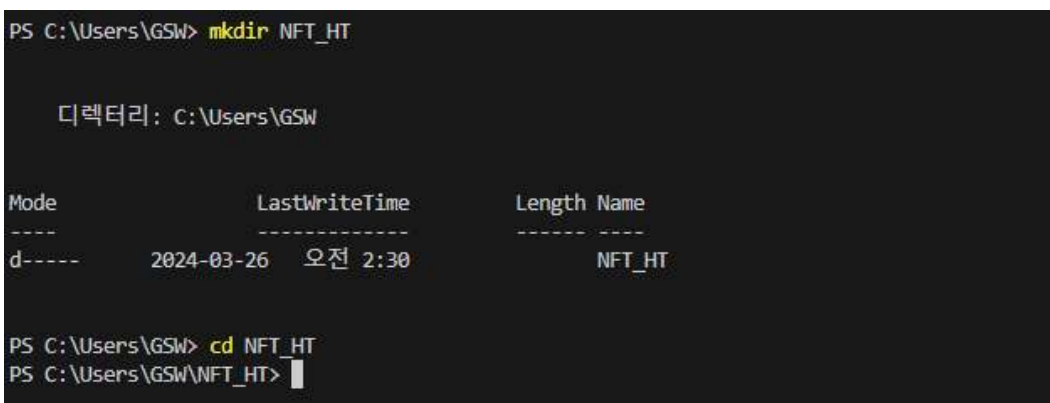
## (2) Intall to Compile

먼저 이번 시간엔 node.js로 프로젝트를 생성할 예정이므로 node.js를 설치해야 한다. node.js를 설치하는 과정은 따로 설명하지 않겠다. node.js의 설치를 완료했다면 이제부터 본격적으로 Hardhat을 설치, 사용하는 방법을 설명하겠다.

1. cmp 프롬프트 창에서 code .을 입력하여 VS Code를 띄우고 터미널창을 열어준다.



2. 프로젝트를 생성할 디렉토리를 만들어주고 생성한 디렉토리로 들어간다.



### 3. npm 프로젝트를 초기화한다.

```
PS C:\Users\GSW\NFT_HT> npm init -y
Wrote to C:\Users\GSW\NFT_HT\package.json:

{
  "name": "nft_ht",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

npm은 자바스크립트 코드의 패키지 관리자이며 npm을 이용해 Hardhat을 설치할 수 있다.

'-y' : yes를 의미하며 프로젝트의 세부 정보를 입력하지 않고 기본 설정으로 프로젝트를 초기화시키는 옵션

### 4. Hardhat 패키지를 개발용 의존성으로 설치한다.

```
PS C:\Users\GSW\NFT_HT> npm install --save-dev hardhat

added 262 packages, and audited 263 packages in 21s

53 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

--save-dev : Node.js 프로젝트에서 사용되는 npm패키지를 개발용 의존성으로 설치시키는 옵션

5. Hardhat을 이용해 새로운 이더리움 스마트 컨트랙트 개발 프로젝트를 초기화한다.

```
PS C:\Users\GSW\NFT_HT> npx hardhat init
888 888 888 888
888 888 888 888
888 888 888 888
8888888888 8888b. 888d888 .d88888 88888b. 8888b. 888888
888 888 "88b 888P" d88" 888 888 "88b "88b 888
888 888 .d888888 888 888 888 888 .d888888 888
888 888 888 888 888 Y88b 888 888 888 888 Y88b.
888 888 "Y888888 888 "Y88888 888 888 "Y888888 "Y888

Welcome to Hardhat v2.22.2

? What do you want to do? ...
> Create a JavaScript project
  Create a TypeScript project
  Create a TypeScript project (with Viem)
  Create an empty hardhat.config.js
  Quit
```

6. 초기화 명령어를 입력하면 위와 같은 화면이 나오는데 'Create an empty hardhat.config.js'를 선택하여 빈 프로젝트를 만들어준다.

```
✓ What do you want to do? · Create an empty hardhat.config.js
Config file created

Give Hardhat a star on Github if you're enjoying it!

https://github.com/NomicFoundation/hardhat
```

7. 사용할 패키지들을 개발용 의존성으로 설치해준다.

```
PS C:\Users\GSW\NFT_HT> npm install --save-dev @nomicfoundation/hardhat-toolbox
added 299 packages, and audited 562 packages in 57s

93 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\GSW\NFT_HT> npm install --save-dev @openzeppelin/hardhat-upgrades
added 106 packages, and audited 668 packages in 11s

138 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\GSW\NFT_HT> npm install --save-dev @openzeppelin/contracts
added 1 package, and audited 669 packages in 2s

138 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

## 설치한 패키지

- **@nomicfoundation/hardhat-toolbox** : hardhat 프레임워크를 보완하는 데 사용되는 도구 모음으로 스마트 컨트랙트 개발과 관리를 더 효율적으로 만들어준다.
- **@openzeppelin/hardhat-upgrades** : Hardhat 프로젝트에서 스마트 컨트랙트의 업그레이드를 관리하는데 사용된다. 컨트랙트를 개발하는 동안 업그레이드 가능한 컨트랙트를 쉽게 작성하고 배포할 수 있도록 도와준다.
- **@openzeppelin/contracts** : 안전한 스마트 컨트랙트를 빠르게 작성하고 배포하기 위한 라이브러리 모음으로 다양한 스마트 컨트랙트 패턴 및 유용한 라이브러리가 포함되어있다.

8. 생성된 `hardhat.config.js`로 들어가 컴파일러를 우리가 사용할 Solidity 컴파일러로 바꿔준다.

```

NFT_HT
├── node_modules
├── JS hardhat.config.js
├── {} package-lock.json
├── {} package.json
└──

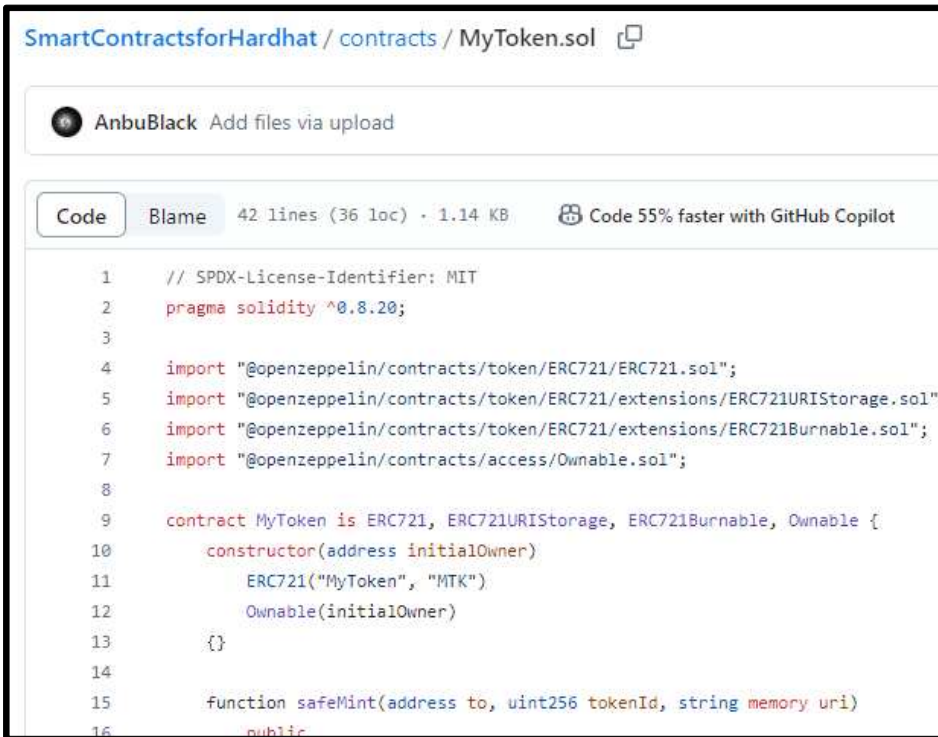
NFT_HT > JS hardhat.config.js > ...
1  require("@nomicfoundation/hardhat-toolbox");
2  require('@openzeppelin/hardhat-upgrades');
3  /** @type import('hardhat/config').HardhatUserConfig */
4  module.exports = {
5    solidity: "0.8.20",
6  };

```

9. contracts 디렉토리를 생성하고 MyToken.sol 파일을 만든다.

<https://github.com/AnbuBlack/SmartContractsforHardhat/blob/main/contracts/MyToken.sol>

MyToken.sol의 소스코드는 위의 링크에 있다.

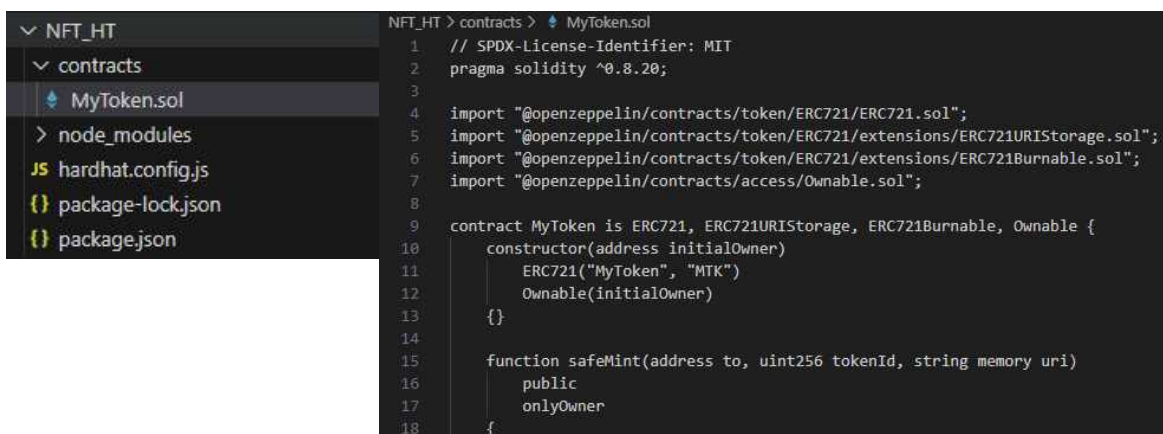


```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.20;
3
4 import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
5 import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
6 import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Burnable.sol";
7 import "@openzeppelin/contracts/access/Ownable.sol";
8
9 contract MyToken is ERC721, ERC721URIStorage, ERC721Burnable, Ownable {
10     constructor(address initialOwner)
11         ERC721("MyToken", "MTK")
12         Ownable(initialOwner)
13     {}
14
15     function safeMint(address to, uint256 tokenId, string memory uri)
16         public

```

contracts 디렉토리 밑에 MyToken.sol을 생성한 모습



```

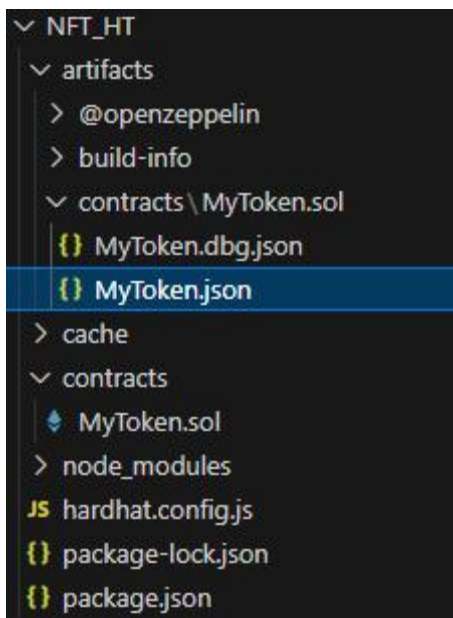
NFT_HT > contracts > MyToken.sol
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.20;
3
4 import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
5 import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
6 import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Burnable.sol";
7 import "@openzeppelin/contracts/access/Ownable.sol";
8
9 contract MyToken is ERC721, ERC721URIStorage, ERC721Burnable, Ownable {
10     constructor(address initialOwner)
11         ERC721("MyToken", "MTK")
12         Ownable(initialOwner)
13     {}
14
15     function safeMint(address to, uint256 tokenId, string memory uri)
16         public
17         onlyOwner
18     {

```

## 10. 다시 NFT\_HT 디렉토리로 돌아간 후 컴파일 실행

```
PS C:\Users\GSW\NFT_HT\contracts> cd ..  
PS C:\Users\GSW\NFT_HT> npx hardhat compile  
✓ Help us improve Hardhat with anonymous crash reports & basic usage data? (Y/n) · y  
Downloading compiler 0.8.24  
Compiled 18 Solidity files successfully (evm target: paris).
```

컴파일을 하고 나면 새로운 디렉토리가 생성된 걸 확인할 수 있는데 여기서 artifact/contracts\MyToken.sol로 들어가보면 MyToken.json 파일이 생성되어있는 것을 볼 수 있다.



이 파일을 이용해 deploy를 진행할 수 있는데 그 과정은 다음 시간에 배우게 될 예정이다.

## 5. 실습 결론 및 소감

이름	내용
김창오	<p>이번 주는 단순히 Metamask를 활용하고 NFT를 발행하는 것을 넘어 직접 스마트 컨트랙트를 짜고 컴파일하는 과정을 배웠다.</p> <p>처음으로 Pinata를 활용하는 방법을 배웠는데, JPEG파일과 JSON파일만 있으면 NFT를 발행할 수 있어서 신기했다.</p> <p>다음으로 Hardhat 프레임워크를 활용해 Node.js환경에서 스마트 컨트랙트를 다루는 방법을 배웠다.Compile과정까지만 배웠지만 꽤나 복잡한 내용이 많았다. 특히 Node.js환경 자체에 어색함이 있어 어려움을 많이 겪었다. 다음 주 예정되었는 Deploy를 배우기 전에 많은 학습이 필요할 것 같다.</p> <p>프로젝트가 NFT를 민팅할 수 있는 웹사이트 개념을 가지므로 앞으로 배울 내용이 더욱 기대된다.</p>
최원제	<p>1. Pinata에 이미지와 JSON 파일을 업로드해 NFT 코인 발행하기</p> <p>2. Node.js와 hardhat을 사용하여 스마트 컨트랙트 개발 환경 만들기</p> <p>전자의 경우 단순히 Pianta에 파일을 업로드 한 뒤 이전에 진행했던 NFT 발급 과정을 진행하면 되었기에 큰 문제가 없었다. 하지만 후자의 경우 본인이 Node.js를 사용해 보는 것이 처음 이였고, 이로 인해 여러 오류들이 발생하여 어려움이 있었다.</p> <p>따라서 원활한 실습 및 프로젝트 개발을 위해서는 Node.js에 대한 공부가 필요하다고 느꼈다. 한편 해당 실습을 통해 로컬 환경에서 스마트 컨트랙트를 개발하고 테스트할 수 있는 환경을 구축할 수 있게 되었다. 이를 통해 이후에 진행할 NFT 웹 서비스의 구조를 보다 구체적으로 구상할 수 있었다.</p>
구선우	<p>이번 시간에는 Pinata를 이용해 json 파일과 이미지 파일을 IPFS 형태로 저장하고 그 파일을 이용해 NFT를 생성하는 방법과 node.js를 이용해 hardhat을 설치하고 hardhat 프로젝트를 생성하고 컴파일하는 과정까지 배웠다.</p> <p>IPFS에 대한 개념이 약한 상태에서 생소한 도구들을 사용해 실습을 진행하니 헛갈리기도 하고 뭐가 어떻게 돌아가는지 잘 모르곤했었다. 그래서 이번 주 보고서를 맡아 IPFS 등에 대한 기본적인 개념들부터 학습을 하고 나니 실습에서 뭐가 왜 안 됐었는지 무엇을 해야하는지 알게 되었다. 어렵פות이 알고 있던 것들을 확실하게 이해하고 나니 어려웠던 부분들을 잘 해결할 수 있게 되었고 흥미도 붙일 수 있게 되었다.</p>
강은솔	<p>지난 주가 NFT 토큰을 어떤 방식으로 발행을 하는지에 대해 주로 배웠다면 이번 주는 이러한 NFT의 요소들이 어떻게 구성되는지, 또 Hardhat을 이용한 컴파일 방식을 알 수 있었다.</p> <p>지금까지 사용해왔던 익숙한 도구들이 아닌 Pinata나 Hardhat을 사용하는 방식이 조금 낯설고 어려운 부분도 꽤나 있었다. 특히, Hardhat의 경우 필요한 패키지들도 상당수 있었고 운영체제 환경마다 명령어가 다른 부분이 특히나 헛갈리는 부분이었다.</p> <p>하지만 직접 JPG파일을 통해 NFT를 생성하는 과정은 흥미로웠고 잘 알지못했던 개념들을 천천히 이해한 후 실습을 진행하니 어렵던 부분들도 해결할 수 있게 되었다.</p>